

PLANAR- \mathcal{F} DELETION: Approximation and Optimal FPT Algorithms

FEDOR V. FOMIN *

DANIEL LOKSHTANOV[†]NEELDHARA MISRA[‡]SAKET SAURABH[§]

Abstract

Let \mathcal{F} be a finite set of graphs. In the \mathcal{F} -DELETION problem, we are given an n -vertex, m -edge graph G and an integer k as input, and asked whether at most k vertices can be deleted from G such that the resulting graph does not contain a graph from \mathcal{F} as a minor. \mathcal{F} -DELETION is a generic problem and by selecting different sets of forbidden minors \mathcal{F} , one can obtain various fundamental problems such as VERTEX COVER, FEEDBACK VERTEX SET or TREewidth η -DELETION.

In this paper we obtain a number of generic algorithmic results about \mathcal{F} -DELETION, when \mathcal{F} contains at least one planar graph. The highlights of our work are

- A randomized $O(nm)$ time constant factor approximation algorithm for the optimization version of \mathcal{F} -DELETION.
- A randomized $O(2^{O(k)}n)$ parameterized algorithm for \mathcal{F} -DELETION when \mathcal{F} is connected. Here a family \mathcal{F} is called connected if every graph in \mathcal{F} is connected. The algorithm can be made deterministic at the cost of making the polynomial factor in the running time $n \log^2 n$ rather than linear.

These algorithms unify, generalize, and improve over a multitude of results in the literature. Our main results have several direct applications, but also the methods we develop on the way have applicability beyond the scope of this paper. Our results – constant factor approximation and FPT algorithms – are stringed together by a common theme of polynomial time preprocessing.

1 Introduction

Let \mathcal{G} be the set of all finite undirected graphs and let \mathcal{L} be the family of all finite subsets of \mathcal{G} . Thus every element $\mathcal{F} \in \mathcal{L}$ is a finite set of graphs and throughout the paper we assume that \mathcal{F} is explicitly given. In this paper we study the following p - \mathcal{F} -DELETION problem.

p - \mathcal{F} -DELETION

Parameter: k

Input: A graph G and a non-negative integer k .

Question: Does there exist $S \subseteq V(G)$, $|S| \leq k$, such that $G \setminus S$ contains no graph from \mathcal{F} as a minor?

The p - \mathcal{F} -DELETION problem defines a wide subclass of node (or vertex) removal problems studied from the 1970s. By the classical theorem of Lewis and Yannakakis [33], deciding

*University of Bergen, Norway. fomin@ii.uib.no

[†]University of California, USA. dlokshtanov@cs.ucsd.edu

[‡]Institute of Mathematical Sciences, India. neeldhara@imsc.res.in

[§]Institute of Mathematical Sciences, India. saket@imsc.res.in

if removing at most k vertices results with a subgraph with property π is NP-complete for every non-trivial property π . By a celebrated result of Robertson and Seymour, every $p\text{-}\mathcal{F}$ -DELETION problem is non-uniformly fixed-parameter tractable (FPT). That is, for every k there is an algorithm solving the problem in time $O(f(k) \cdot n^3)$ [41]. The importance of the result comes from the fact that it simultaneously gives FPT algorithms for a variety of important problems such as VERTEX COVER, FEEDBACK VERTEX SET, VERTEX PLANARIZATION, etc. It is conceivable that meta theorems for vertex deletion problems might be formulated by addressing problems that are expressible in logics such as first order and monadic second order. However, since these capture problems that are known to be intractable, for example INDEPENDENT SET or DOMINATING SET, we do not expect to have a theorem that guarantees tractability for vertex deletion problems through this route. Therefore, the systematic study of the $p\text{-}\mathcal{F}$ -DELETION problems is the more promising way forward to obtain meta-theorems for vertex removal problems on general undirected graphs.

In this paper we show that when $\mathcal{F} \in \mathcal{L}$ contains at least one planar graph, it is possible to obtain a number of generic results advancing known tractability borders of $p\text{-}\mathcal{F}$ -DELETION. The case when \mathcal{F} contains a planar graph, while being considerably more restricted than the general case, already encompasses a number of the well-studied instances of $p\text{-}\mathcal{F}$ -DELETION. For example, when $\mathcal{F} = \{K_2\}$, a complete graph on two vertices, this is the VERTEX COVER problem. When $\mathcal{F} = \{C_3\}$, a cycle on three vertices, this is the FEEDBACK VERTEX SET problem. Another fundamental problem, which is a special case of $p\text{-}\mathcal{F}$ -DELETION, is TREewidth η -DELETION or η -TRANSVERSAL which is to delete at most k vertices to obtain a graph of treewidth at most η . Since any graph of treewidth η excludes a $(\eta + 1) \times (\eta + 1)$ grid as a minor, we have that the set \mathcal{F} of forbidden minors of treewidth η graphs contains a planar graph. TREewidth η -DELETION plays important role in generic efficient polynomial time approximation schemes based on Bidimensionality Theory [25, 26]. Among other examples of $p\text{-}\mathcal{F}$ -DELETION that can be found in the literature on approximation and parameterized algorithms, are the cases of \mathcal{F} being $\{K_{2,3}, K_4\}$, $\{K_4\}$, $\{\theta_c\}$, and $\{K_3, T_2\}$, which correspond to removing vertices to obtain an outerplanar graph, a series-parallel graph, a diamond graph, and a graph of pathwidth one, respectively.

We call a family $\mathcal{F} \in \mathcal{F}$ connected if every graph in \mathcal{F} is connected. The main algorithmic contributions of our work is the following set of results for $p\text{-}\mathcal{F}$ -DELETION for the case that \mathcal{F} contains a planar graph:

- A randomized $O(nm)$ time constant factor approximation algorithm for the optimization version of \mathcal{F} -DELETION.
- A randomized linear time and single exponential parameterized algorithm for $p\text{-}\mathcal{F}$ -DELETION when \mathcal{F} is connected. That is, an algorithm running in time $O(2^{O(k)}n)$. The algorithm can be made deterministic at the cost of making the running time $O(2^{O(k)}n \log^2(n))$ rather than $O(2^{O(k)}n)$.

We use \mathcal{F} to denote the subclass of \mathcal{L} such that every $\mathcal{F} \in \mathcal{F}$ contains a planar graph. Let us remark that for most interesting minor closed graph classes, the set \mathcal{F} of forbidden minors is connected. Specifically, if a graph class Π has the property that a graph G is in Π whenever all G 's connected components are, then the set of forbidden minors to Π is connected.

Methodology. All our results – constant factor approximation and FPT algorithms for $p\text{-}\mathcal{F}$ -DELETION – have a common theme of polynomial time preprocessing. Preprocessing as a strategy for coping with hard problems is universally applied in practice and the notion of *kernelization* in parameterized complexity provides a mathematical framework for analyzing the quality of preprocessing strategies. In parameterized complexity each problem instance

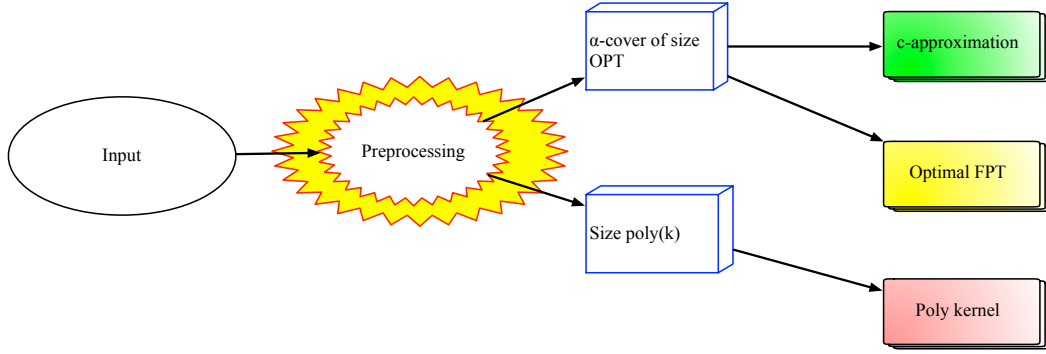


Figure 1: General view of our approach

comes with a parameter k and a central notion in parameterized complexity is *fixed parameter tractability (FPT)*. This means, for a given instance (x, k) , solvability in time $f(k) \cdot p(|x|)$, where f is an arbitrary function of k and p is a polynomial in the input size. The parameterized problem is said to admit a *polynomial kernel* if there is a polynomial time algorithm (the degree of polynomial is independent of k), called a *kernelization* algorithm, that reduces the input instance down to an instance with size bounded by a polynomial $p(k)$ in k , while preserving the answer.

Thus the goal of kernelization is to apply reduction rules such that the size of the reduced instance can be upper bounded by a function of the parameter. However, if we want to use preprocessing for approximation or FPT algorithms, it is not necessary that the size of the reduced instance has to be upper bounded. What we need is a preprocessing procedure that allows us to navigate the solution search space efficiently. Our first contribution is a notion of preprocessing that is geared towards approximation and FPT algorithms. This notion relaxes the demands of kernelization and thus it is possible that a larger set of problems may admit this simplification procedure, when compared to kernelization. For approximation and FPT algorithms, we use the notion of α -cover as a measure of good preprocessing. For $0 < \alpha \leq 1$, we say that a vertex subset $S \subseteq V(G)$ is an α -cover, if the sum of vertex degrees $\sum_{v \in S} d(v)$ is at least $2\alpha|E(G)|$. For example, every vertex cover of a graph is also a 1-cover. The defining property of this preprocessing is that the equivalent simplified instance of the problem admits some optimal solution which is also an α -cover. If we succeed with this goal, then for an edge selected uniformly at random, with a constant probability at least one of its endpoints belong to some optimal solution. Using this as a basic step, we can construct approximation and FPT algorithms. But how to achieve this kind of preprocessing?

To achieve our goals we use the idea of graph replacement dating back to Fellows and Langston [21]. Precisely, what we use is the modern notion of “protrusion reduction” that has been recently employed in [7, 27] for obtaining meta-kernelization theorems for problems on sparse graphs like planar graphs, graphs of bounded genus [8], graphs excluding a fixed graph as a minor or induced subgraph [27, 23], or graphs excluding a fixed graph as a topological minor [32]. In this method, we find a large protrusion – a graph of small treewidth and small boundary – and then the preprocessing rule replaces this protrusion by a protrusion of constant size. One repeatedly applies this until no longer possible. Finally, by using combinatorial arguments one upper bounds the size of the reduced induced (a graph without large protrusion). The FPT algorithms use the replacement technique developed in [8, 23], while for approximation algorithm we need another type of protrusion reduction. The reason

why the normal protrusion replacement does not work for approximation algorithms is the same as why the NP-hardness reduction is not always an approximation preserving reduction. While the normal protrusion replacement works fine for preserving exact solutions, we needed a notion of protrusion reduction that also preserves approximate solutions. To this end, we develop a new notion of *lossless protrusion reduction*, and show that several problems do admit lossless protrusion reductions. We exemplify the usefulness of the new concept by obtaining constant factor approximation algorithms for \mathcal{F} -DELETION. These FPT and approximation algorithms are obtained by showing that solutions to the instances of the problem that do not contain protrusion form an α -cover for some fixed constant α .

As far as we are equipped with new tools and concepts: α -cover and lossless protrusion reduction, we are able to proceed with algorithms for p - \mathcal{F} -DELETION. These algorithms unify and generalize a multitude of results in the literature. In what follows we survey earlier results in each direction and discuss our results.

Approximation. In the optimization version of p - \mathcal{F} -DELETION, we want to compute the minimum set S , which removal leaves input graph G \mathcal{F} -minor-free. We denote this optimization problem by \mathcal{F} -DELETION. Characterising graph properties for which the corresponding vertex deletion problem can be approximated within a constant factor is a long standing open problem in approximation algorithms [43]. In spite of long history of research, we are still far from a complete understanding. Constant factor approximation algorithms for VERTEX COVER are known since 1970s [36, 2]. Lund and Yannakakis observed that the vertex deletion problem for any hereditary property with a finite number of minimal forbidden subgraphs can be approximated with a constant ratio [34]. They also conjectured that for every nontrivial, hereditary property with an infinite number of minimal forbidden subgraphs, the vertex deletion problem cannot be approximated with constant ratio. However, it appeared later that FEEDBACK VERTEX SET admits a constant factor approximation [3, 1] and thus the dividing line of approximability lies somewhere else. On a related matter, Yannakakis [42] showed that approximating the number of vertices to delete in order to obtain *connected* graph with some property π within factor $n^{1-\varepsilon}$ is NP-hard, see [42] for the definition of the property π . This result holds for very wide class of properties, in particular for properties being acyclic and outerplanar. There was no much progress on approximability/non-approximability of vertex deletion problems until recent work of Fiorini et al. [22] who gave a constant factor approximation algorithm for p - \mathcal{F} -DELETION for the case when \mathcal{F} is a diamond graph, i.e., a graph with two vertices and three parallel edges.

Our first contribution is the theorem stating that every graph property π expressible by a finite set of forbidden minors containing at least one planar graph, the vertex deletion problem for property π admits a constant factor approximation algorithm. In other words, we prove the following theorem

Theorem 1. *For every set $\mathcal{F} \in \mathcal{F}$, \mathcal{F} -DELETION admits a randomized (Monte Carlo) constant ratio approximation algorithm with running time $O(nm)$.*

Let us remark that for all known constant factor approximation algorithms of vertex deletion to a hereditary property π , property π is either characterized by an finite number of minimal forbidden subgraphs or by finite number of forbidden minors, one of which is planar. Theorem 1 together with the result of Lund and Yannakakis, not only encompass all known vertex deletion problems with constant factor approximation ratio but significantly extends known tractability borders for such types of problems.

Fast FPT Algorithms. The study of parameterized problems proceeds in several steps. The first step is to establish if the problem on hands is fixed parameter tractable or not. If

the problem is in FPT, then the next steps are to identify if the problem admits a polynomial kernel and to find the fastest possible FPT algorithm solving the problem. The running time of every FPT algorithm is $O(f(k)n^c)$, that is, the product of a super-polynomial function $f(k)$ depending only on the parameter k and polynomial n^c , where n is the input size and c is some constant. Both steps, minimizing super-polynomial function $f(k)$ and minimizing the exponent c of the polynomial part, are important parts in the design and analysis of parameterized algorithms.

The $p\text{-}\mathcal{F}\text{-DELETION}$ problem was introduced by Fellows and Langston [20], who gave a non-constructive algorithm running in time $O(f(k) \cdot n^2)$ for some function $f(k)$ [20, Theorem 6]. This result was improved by Bodlaender [5] to $O(f(k) \cdot n)$, for $f(k) = 2^{2^{O(k \log k)}}$. There is a substantial amount of work on improving the exponential function $f(k)$ for special cases of $p\text{-}\mathcal{F}\text{-DELETION}$. For the VERTEX COVER problem the existence of single-exponential algorithms is well-known since almost the beginnings of the field of Parameterized Complexity, the current best algorithm being by Chen et al. [14]. Randomized parameterized single exponential algorithm for FEEDBACK VERTEX SET was given by Becker et al. [4] but existence of deterministic single-exponential algorithms for FEEDBACK VERTEX SET was open for a while and it took some time and discovery of iterative compression [39] to reduce the running time to $2^{O(k)}n^{O(1)}$ [11, 13, 17, 19, 28, 38]. The current champion for FEEDBACK VERTEX SET are the deterministic algorithm of Cao et al. [11] with running time $O(3.83^k kn^2)$ and the randomized of Cygan et al. with running time $3^k n^{O(1)}$ [17]. Recently, Joret et al. [30] showed that $p\text{-}\mathcal{F}\text{-DELETION}$ for $\mathcal{F} = \{\theta_c\}$, where θ_c is the graph with two vertices and c parallel edges, can be solved in time $2^{O(k)}n^{O(1)}$ for every fixed c . Philip et al. [37] studied PATHWIDTH 1-DELETION and obtained an algorithm with running time $O(7^k n^2)$ that was later improved to $O(4.65^k n^{O(1)})$ in [18]. Kim et al. [31] gave a single exponential algorithm for $\mathcal{F} = \{K_4\}$. Unless Exponential Time Hypothesis (ETH) fails [12, 29], single exponential dependence on the parameter k is asymptotically the best bound one can obtain for $p\text{-}\mathcal{F}\text{-DELETION}$, and thus our next theorem provides asymptotically optimal bounds on the exponential function of the parameter and polynomial contribution of the input.

Theorem 2. *For every connected set $\mathcal{F} \in \mathcal{F}$ containing a planar graph, there is a randomized (Monte Carlo) algorithm solving $p\text{-}\mathcal{F}\text{-DELETION}$ in time $O(c^k n)$ for some constant $c > 1$.*

We finally give a deterministic algorithm for $p\text{-}\mathcal{F}\text{-DELETION}$. Surprisingly, our algorithm does not use iterative compression but is based on branching on degree sequences.

Theorem 3. *For every connected set $\mathcal{F} \in \mathcal{F}$ containing a planar graph, $p\text{-}\mathcal{F}\text{-DELETION}$ is solvable in time $O(c^k n \log^2 n)$ for some constant $c > 1$.*

2 Preliminaries

In this section we give various definitions which we use in the paper. We use $V(G)$ to denote the vertex set of a graph G , and $E(G)$ to denote the edge set. The degree of a vertex v in G is the number of edges incident on v , and is denoted by $d(v)$. A graph G' is a *subgraph* of G if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. The subgraph G' is called an *induced subgraph* of G if $E(G') = \{\{u, v\} \in E(G) \mid u, v \in V(G')\}$. Given a subset $S \subseteq V(G)$ the subgraph induced by S is denoted by $G[S]$. The subgraph induced by $V(G) \setminus S$ is denoted by $G \setminus S$. We denote by $N_G(S)$ the open neighborhood of S , i.e. the set of vertices in $V(G) \setminus S$ adjacent to S . Whenever the graph G is clear from the context, we omit the subscript in $N_G(S)$ and denote it only by $N(S)$. By $N[S]$ we denote $N(S) \cup S$. Let \mathcal{F} be a finite set of graphs. A vertex

subset $S \subseteq V(G)$ of a graph G is said to be a \mathcal{F} -deletion set if $G \setminus S$ does not contain any graphs in the family \mathcal{F} as a minor.

2.1 Parameterized algorithms and kernels.

A parameterized problem Π is a subset of $\Gamma^* \times \mathbb{N}$ for some finite alphabet Γ . An instance of a parameterized problem consists of (x, k) , where k is called the parameter. We assume that k is given in unary and hence $k \leq |x|$. A central notion in parameterized complexity is *fixed parameter tractability (FPT)* which means, for a given instance (x, k) , solvability in time $f(k) \cdot p(|x|)$, where f is an arbitrary function of k and p is a polynomial in the input size. The notion of *kernelization* is formally defined as follows.

Definition 1. [Kernelization] Let $\Pi \subseteq \Gamma^* \times \mathbb{N}$ be a parameterized problem and g be a computable function. We say that Π admits a kernel of size g if there exists an algorithm K , called kernelization algorithm, or, in short, a kernelization, that given $(x, k) \in \Gamma^* \times \mathbb{N}$, outputs, in time polynomial in $|x| + k$, a pair $(x', k') \in \Gamma^* \times \mathbb{N}$ such that

- (a) $(x, k) \in \Pi$ if and only if $(x', k') \in \Pi$, and
- (b) $\max\{|x'|, k'\} \leq g(k)$.

When $g(k) = k^{O(1)}$ or $g(k) = O(k)$ then we say that Π admits a polynomial or linear kernel respectively. If additionally $k' \leq k$ we say that the kernel is strict.

2.2 Treewidth.

Let G be a graph. A *tree decomposition* of G is a pair $T, \mathcal{X} = \{X_t\}_{t \in V(T)}$ where T is a tree and \mathcal{X} is a collection of subsets of $V(G)$ such that:

- $\forall e = uv \in E(G), \exists t \in V(T) : \{u, v\} \subseteq X_t$ and
- $\forall v \in V(G), T[\{t \mid v \in X_t\}]$ is a non-empty connected subtree of T .

We call the vertices of T *nodes* and the sets in \mathcal{X} *bags* of the tree decomposition (T, \mathcal{X}) . The *width* of (T, \mathcal{X}) is equal to $\max\{|X_t| - 1 \mid t \in V(T)\}$ and the *treewidth* of G is the minimum width over all tree decompositions of G .

A *nice tree decomposition* is a pair (T, \mathcal{X}) where (T, \mathcal{X}) is a tree decomposition such that T is a rooted tree and the following conditions are satisfied:

- Every node of the tree T has at most two children;
- if a node t has two children t_1 and t_2 , then $X_t = X_{t_1} = X_{t_2}$; and
- if a node t has one child t_1 , then either $|X_t| = |X_{t_1}| + 1$ and $X_{t_1} \subset X_t$ (in this case we call t_1 *insert node*) or $|X_t| = |X_{t_1}| - 1$ and $X_t \subset X_{t_1}$ (in this case we call t_1 *insert node*).

It is possible to transform a given tree decomposition (T, \mathcal{X}) into a nice tree decomposition (T', \mathcal{X}') in time $O(|V| + |E|)$ [6].

2.3 Minors

Given an edge $e = xy$ of a graph G , the graph G/e is obtained from G by contracting the edge e , that is, the endpoints x and y are replaced by a new vertex v_{xy} which is adjacent to the old neighbors of x and y (except from x and y). A graph H obtained by a sequence of edge-contractions is said to be a *contraction* of G . We denote it by $H \leq_c G$. A graph H is a *minor* of a graph G if H is the contraction of some subgraph of G and we denote it by $H \leq_m G$. We say that a graph G is *H -minor-free* when it does not contain H as a minor. We also say that a graph class \mathcal{G} is *H -minor-free* (or, excludes H as a minor) when all its members are H -minor-free. It is well-known [40] that if $H \leq_m G$ then $tw(H) \leq tw(G)$. We will also use the following fact about excluding planar graphs as minors.

Proposition 1. *There is a constant c such that for every planar H and graph G with $tw(G) \geq 2^{c|V(H)|^3}$, H is a minor of G .*

2.4 t -Boundaried graphs and Gluing.

A t -boundaried graph is a graph G and a set $B \subset V(G)$ of size at most t with each vertex $v \in B$ having a label $\ell_G(v) \in \{1, \dots, t\}$. Each vertex in B has a unique label. We refer to B as the *boundary* of G . For a t -boundaried G the function $\delta(G)$ returns the boundary of G . Two t -boundaried graphs G and H are isomorphic if there is a bijection f from $V(G)$ to $V(H)$ such that $uv \in E(G) \iff f(u)f(v) \in E(H)$, for every $v \in \delta(G)$ we have $f(v) \in \delta(H)$ and $\ell_G(v) = \ell_H(f(v))$. Specifically f is an isomorphism between G and H in the normal graph sense, but additionally f respects the labels of the border vertices. Observe that a t -boundaried graph may have no boundary at all. A graph G is isomorphic to a t -boundaried graph H if there is an isomorphism between G and H .

Two t -boundaried graphs G_1 and G_2 can be *glued* together to form a graph $G = G_1 \oplus G_2$. The gluing operation takes the disjoint union of G_1 and G_2 and identifies the vertices of $\delta(G_1)$ and $\delta(G_2)$ with the same label. If there are vertices $u_1, v_1 \in \delta(G_1)$ and $u_2, v_2 \in \delta(G_2)$ such that $\ell_{G_1}(u_1) = \ell_{G_2}(u_2)$ and $\ell_{G_1}(v_1) = \ell_{G_2}(v_2)$ then G has vertices u formed by unifying u_1 and u_2 and v formed by unifying v_1 and v_2 . The new vertices u and v are adjacent if $u_1v_1 \in E(G_1)$ or $u_2v_2 \in E(G_2)$.

The *boundaried gluing operation* \oplus_δ is similar to the normal gluing operation, but results in a t -boundaried graph rather than a graph. Specifically $G_1 \oplus_\delta G_2$ results in a t -boundaried graph where the graph is $G = G_1 \oplus G_2$ and a vertex is in the boundary of G if it was in the boundary of G_1 or G_2 . Vertices in the boundary of G keep their label from G_1 or G_2 . Both for gluing and boundaried gluing we will refer to $G_1 \oplus G_2$ or $G_1 \oplus_\delta G_2$ as the *sum* of G_1 and G_2 , and G_1 and G_2 are the *terms* of the sum.

For a t -boundaried graph G and boundary vertex $v \in \delta(G)$, *forgetting* v results in a t -boundaried graph identical to G , except that v is no longer a boundary vertex. All other boundary vertices keep their labels. Forgetting a non-boundary vertex leaves the graph unchanged, as does forgetting a vertex that is not in the vertex set of G . Forgetting a set $S \subseteq \delta(G)$ of vertices means forgetting all vertices in the set. The function **forget**(G, S) returns the t -boundaried graph resulting from forgetting S in G .

We will frequently need to construct t -boundaried graphs from subgraphs of a graph G . For a graph G and two disjoint vertex sets P and B we define G_P^B to be the t -boundaried graph $G[P \cup B]$ with boundary B . The labelling of the border B is chosen in a manner independent of P - such that if P_1, P_2 and B are disjoint then $G_{P_1}^B \oplus_\delta G_{P_2}^B = G_{P_1 \cup P_2}^B$.

2.5 Monadic Second Order Logic (MSO)

The syntax of MSO on graphs includes the logical connectives $\vee, \wedge, \neg, \Leftrightarrow, \Rightarrow$, variables for vertices, edges, sets of vertices and sets of edges, the quantifiers \forall, \exists that can be applied to these variables, and the following five binary relations:

1. $u \in U$ where u is a vertex variable and U is a vertex set variable;
2. $d \in D$ where d is an edge variable and D is an edge set variable;
3. $\text{inc}(d, u)$, where d is an edge variable, u is a vertex variable, and the interpretation is that the edge d is incident on the vertex u ;
4. $\text{adj}(u, v)$, where u and v are vertex variables u , and the interpretation is that u and v are adjacent;
5. equality of variables representing vertices, edges, set of vertices and set of edges.

Many common graph-theoretic notions such as vertex degree, connectivity, planarity, being acyclic, and so on, can be expressed in MSO, as can be seen from introductory expositions [9, 15].

H minor-models. Recall that a t -boundaried graph H is a minor of a t -boundaried graph G if (a t -boundaried graph isomorphic to) H can be obtained from G by deleting vertices or edges or contracting edges, but never contracting edges with both endpoints being boundary vertices. Let $V(H) = \{h_1, \dots, h_c\}$, and let $B_G := \{b_1^G, \dots, b_t^G\}$ and $B_H := \{b_1^H, \dots, b_t^H\}$ denote $\delta(G)$ and $\delta(H)$ respectively. Then, the formulation that $H \leq_m G$ is given by $\phi(G, H, B_G, B_H)$:

$$\begin{aligned} \phi(G, H, B_G, B_H) \equiv \exists X_1, \dots, X_c \subseteq V(G) [& \\ & \bigwedge_{i \neq j} (X_i \cap X_j = \emptyset) \wedge \bigwedge_{1 \leq i \leq c} \text{Conn}(G, X_i) \wedge \\ & \bigwedge_{(h_i, h_j) \in E(H)} \exists x \in X_i \wedge y \in X_j [(x, y) \in E(G)] \wedge \\ & \bigwedge_{(b_i^H \in B_H)} \exists x \in X_i [x = b_i^G] \\ &] \end{aligned} \quad (1)$$

2.6 Finite Integer Index and Protrusions

For a parameterized problem Π and two t -boundaried graphs $G_1, G_2 \in \mathcal{G}$, we say that $G_1 \equiv_\Pi G_2$ if there exists a constant c such that for every t -boundaried graph G and for every integer k , $(G_1 \oplus G, k) \in \Pi$ if and only if $(G_2 \oplus G, k + c) \in \Pi$. For every t , the relation \equiv_Π on t -boundaried graphs is an equivalence relation, and we call \equiv_Π the *canonical equivalence relation* of Π . We say that a problem Π has *Finite Integer Index* if for every t , \equiv_Π has finite index on t -boundaried graphs. Thus, if Π has finite integer index then for every t there is a finite set \mathcal{S} of t -boundaried graphs for every t -boundaried graph G_1 there exists $G_2 \in \mathcal{S}$ such that $G_2 \equiv_\Pi G_1$. Such a set \mathcal{S} is called a *set of representatives for* (Π, t) . We will repeatedly make use of the following proposition.

Proposition 2 ([8]). *For every connected $\mathcal{F} \in \mathcal{F}$, \mathcal{F} -DELETION has finite integer index.*

Protrusions and Protrusion Replacement For a graph G and $S \subseteq V(G)$, we define $\partial_G(S)$ as the set of vertices in S that have a neighbor in $V(G) \setminus S$. For a set $S \subseteq V(G)$ the *neighbourhood* of S is $N_G(S) = \partial_G(V(G) \setminus S)$. When it is clear from the context, we omit the subscripts. A r -*protrusion* in a graph G is a set $X \subseteq V$ such that $|\partial(X)| \leq r$ and $\text{tw}(G[X]) \leq r$. If G is a graph containing a r -protrusion X and X' is a r -boundaried graph, the act of *replacing* X by X' means replacing G by $G_{V(G) \setminus X}^{\partial(X)} \oplus X'$.

A *protrusion replacer* for a parameterized graph problem Π is a family of algorithms, with one algorithm for every constant r . The r 'th algorithm has the following specifications. There exists a constant r' (which depends on r) such that given an instance (G, k) and an r -protrusion X in G of size at least r' , the algorithm runs in time $O(|X|)$ and outputs an instance (G', k') such that $(G', k') \in \Pi$ if and only if $(G, k) \in \Pi$, $k' \leq k$ and G' is obtained from G by replacing X by a r -boundaried graph X' with less than r' vertices. Observe that since X has at least r' vertices and X' has less than r' vertices this implies that $|V(G')| < |V(G)|$. The following proposition is the driving force of [8] and the starting point for our algorithms.

Proposition 3 ([8]). *Every parameterized problem with finite integer index has a protrusion replacer.*

Together, Propositions 2 and 3 imply that for every connected $\mathcal{F} \in \mathcal{F}$, \mathcal{F} -DELETION has a protrusion replacer.

2.6.1 Least Common Ancestor-Closure of Sets in Trees.

For a rooted tree T and vertex set M in $V(T)$ the least common ancestor-closure (*LCA-closure*) $\mathbf{LCA-closure}(M)$ is obtained by the following process. Initially, set $M' = M$. Then, as long as there are vertices x and y in M' whose least common ancestor w is not in M' , add w to M' . When the process terminates, output M' as the LCA-closure of M . The following folklore lemma summarizes two basic properties of LCA closures.

Lemma 1. *Let T be a tree, $M \subseteq V(T)$ and $M' = \mathbf{LCA-closure}(M)$. Then $|M'| \leq 2|M|$ and for every connected component C of $T \setminus M'$, $|N(C)| \leq 2$.*

Proof. To prove that $|M'| \leq 2|M|$ make a tree T' with vertex set M' , and for every vertex $v \in M'$ adding an edge to the lowermost ancestor of v in M' in the tree T . Observe that in T' all leaves are from M , since every vertex in $M' \setminus M$ is the least common ancestor of two vertices below it in T . Furthermore, for the same reason every vertex in $M' \setminus M$ has at least two descendants in T' . A standard counting argument for trees shows that the number of vertices with at least two descendants is at most the number of leaves. Hence $|M' \setminus M| \leq |M|$ and so $|M'| \leq 2|M|$.

We now prove that $|N(C)| \leq 2$. Suppose not, and let r be the root of P . At most one of C 's neighbours is the parent of r and hence at least two of C 's neighbours, say u and v are children of vertices in C . The vertices u and v are both in M' , and they are both descendants of r . But then the least common ancestor of u and v must lie in C and hence is not in M' , contradicting the construction of M' . So we conclude that $|N(P)| \leq 2$. \square

3 A Randomized Algorithm for “connected” p - \mathcal{F} -DELETION

In this section we give a randomized algorithm for p - \mathcal{F} -DELETION when every graph in $\mathcal{F} \in \mathcal{F}$ is connected. Recall that we call a family \mathcal{F} connected if all the graphs in \mathcal{F} is connected. We will show that for every connected \mathcal{F} the algorithm runs in polynomial time,

with the exponent of the polynomial depending on the family \mathcal{F} . If the input graph has a \mathcal{F} -deletion set of size at most k , the algorithm will detect a \mathcal{F} -deletion set of size at most k with probability at least $\frac{1}{c^k}$. Here the constant c depends on \mathcal{F} . The algorithm has no false positives - we show that if it reports that a \mathcal{F} -deletion set of size at most k exists then G indeed has such a set.

In the following sections we will progressively improve the algorithm; first we give an implementation of the algorithm with expected running time $O(n \cdot OPT)$. Then we show how to modify the (sped up) algorithm so that it not only decides whether G has a \mathcal{F} -deletion set of size at most k , but also outputs a solution. We show that if G has a \mathcal{F} -deletion set of size at most k , the algorithm will output a solution of size k with probability at least $\frac{1}{c^k}$. We then proceed to show that this algorithm in fact outputs constant factor approximate solutions with constant probability, yielding a constant factor approximation for $p\text{-}\mathcal{F}\text{-DELETION}$ for connected \mathcal{F} in expected $O(n \cdot OPT)$ time. The main structure of the improved algorithm remains the same as the one described here.

The first building block of our algorithm is a simple algorithm to reduce the input instance to an equivalent instance that does not contain any large protrusions with small border.

Lemma 2. *For every $\mathcal{F} \in \mathcal{F}$ and constants r and r' such that $p\text{-}\mathcal{F}\text{-DELETION}$ has a protrusion replacer that reduces r -protrusions of size r' , there is an algorithm that takes as input an instance (G, k) of $p\text{-}\mathcal{F}\text{-DELETION}$, runs in $n^{O(r')}$ time and outputs an equivalent instance (G', k') such that $|V(G')| \leq V(G)$, $k' \leq k$ and G' has no r -protrusion of size at least r' .*

Proof. It is sufficient to give a $n^{O(r')}$ time algorithm to find a r -protrusion X in G of size at least r' , if such a protrusion exists. If we had such an algorithm to find a protrusion we could keep looking for r -protrusions X in G of size at least r' , and if one is found replacing them using the protrusion replacer. Since each replacement decreases the number of vertices by one we converge to an instance (G', k') with the desired properties after at most n iterations.

To find an r -protrusion of size at least r' observe that if such a protrusion exists, then there must be at least one such protrusion X such that $G[X \setminus \partial(X)]$ has at most r' connected components. Indeed, if $G[X \setminus \partial(X)]$ has more than r' connected components then let X' be $\partial(X)$ plus the union of any r' components of $G[X \setminus \partial(X)]$. Now X' is an r -protrusion of size at least r' and $G[X' \setminus \partial(X')]$ has at most r' components. To find a r -protrusion X of size at least r' on at most r' components, guess $\partial(X)$ and then guess which components of $G \setminus \partial(X)$ are in X . The size of the search space is bounded by $n^r \cdot n^{r'}$ and for each candidate X we can test whether it is a protrusion in linear time using Bodlaender's linear time treewidth algorithm [6]. \square

The second building block of our algorithm is a lemma whose proof we postpone until the end of this section. The lemma states that for any $\mathcal{F} \in \mathcal{F}$, if G contains no large protrusions with small border then any feasible solution to $p\text{-}\mathcal{F}\text{-DELETION}$ is incident to a linear fraction of the edges of G . Recall that an α -cover in G is a set S such that $\sum_{v \in S} d(v) \geq \alpha \cdot \sum_{v \in V(G)} d(v) = 2\alpha \cdot m$.

Lemma 3. *For every $\mathcal{F} \in \mathcal{F}$ there exist constants r and α such that if a graph G has no r -protrusion of size at least r' , then every \mathcal{F} -deletion set S of G is a $\frac{\alpha}{r'}$ -cover of G .*

We now combine Lemmata 2 and 3 to give a randomized algorithm for $p\text{-}\mathcal{F}\text{-DELETION}$ for all $\mathcal{F} \in \mathcal{F}$ such that each graph in \mathcal{F} is connected.

Lemma 4. *Algorithm 2 runs in polynomial time, if (G, k) is a “no” instance it outputs “no” and if (G, k) is a “yes” instance it outputs “yes” with probability at least $\frac{1}{c^k}$ where c is a constant depending only on \mathcal{F} .*

<p>Randomized-FPT-beta((G, k))</p> <p>Set $G_{\text{current}} := G$ and $k_{\text{current}} := k$.</p> <p>While (G_{current} is not \mathcal{F}-free) do as follows:</p> <ol style="list-style-type: none"> 1. If $k_{\text{current}} \leq 0$ return that G does not have a k-sized \mathcal{F}-deletion set . 2. Apply Lemma 2 on $(G_{\text{current}}, k_{\text{current}})$ and obtain an equivalent instance (G', k'). 3. Pick a vertex $u \in V(G)$ at random with probability $\frac{d(u)}{2m}$. Set $G_{\text{current}} := G' \setminus \{u\}$ and $k_{\text{current}} := k' - 1$ <p>Return that G has a k-sized \mathcal{F}-deletion set .</p>
--

Figure 2: In Algorithm **Randomized-FPT-beta**, let r be the constant as guaranteed by Lemma 3 and let r' be the smallest integer such that the protrusion replacer for \mathcal{F} -DELETION reduces r -protrusions of size r' .

Proof. Since each iteration runs in polynomial time and reduces the number of vertices in G_{current} by at least one, Algorithm 2 runs in polynomial time. Furthermore, Step 2 reduces the instance to an equivalent instance with $k' \leq k_{\text{current}}$ and Step 3 only decreases k_{current} when it puts a vertex into the solution. Hence when the algorithm outputs “yes” then a k -sized \mathcal{F} -deletion set exists. It remains to show the last part of the statement.

We say that an iteration of Step 3 is *successful* if there exists a \mathcal{F} -deletion set S of G' with $|S| \leq k'$ such that the vertex u selected in this step is in S . If the step is successful then $S \setminus \{u\}$ is a \mathcal{F} -deletion set of G' of size at most $k' - 1$. Thus, if the input graph G has a k -sized \mathcal{F} -deletion set and all the iterations of Step 3 are successful then the algorithm maintains the invariant that G_{current} has a \mathcal{F} -deletion set of size at most k_{current} , and thus after at most k iterations it terminates and outputs that (G, k) is a “yes” instance. When Step 3 is executed the graph G' has no r -protrusions of size at least r' . Thus by Lemma 3 every \mathcal{F} -deletion set of G' is an $\frac{\alpha}{r'}$ -cover for a constant α depending only on \mathcal{F} . Hence the probability that u is in a minimum size \mathcal{F} -deletion set of G' is at least $\frac{\alpha}{r'}$. We conclude that the probability that the first k executions of Step 3 are successful is at least $(\frac{\alpha}{r'})^k$ concluding the proof. \square

Repeating the algorithm presented in Figure 2 $O(c^k)$ times yields a $O(2^{O(k)} n^{O(1)})$ time algorithm for $p\mathcal{F}$ -DELETION for all connected $\mathcal{F} \in \mathcal{F}$. However we are not entirely done with the proof of Lemma 4, as it remains to prove Lemma 3. In order to complete the proof we need to define protrusion decompositions.

3.1 Protrusion Decompositions and Proof of Lemma 3

We recall the notion of a protrusion decomposition defined in [8] and show that if a graph G has a set X such that $\mathbf{tw}(G \setminus X) \leq d$, then it admits a protrusion decomposition for an appropriate value of the parameters. We then use this result to prove Lemma 3.

Definition 1. [Protrusion Decomposition][[8]] *A graph G has an (α, β) -protrusion decomposition if $V(G)$ has a partition $\mathcal{P} = \{R_0, R_1, \dots, R_t\}$ where*

- $\max\{t, |R_0|\} \leq \alpha$,
- each $N_G[R_i]$, $i \in \{1, \dots, t\}$ is a β -protrusion of G , and

- for all $i > 1$, $N[R_i] \subseteq R_0$.

We call the sets $R_i^+ = N_G[R_i]$, $i \in \{1, \dots, t\}$ protrusions of \mathcal{P} .

We now show that for every $\mathcal{F} \in \mathcal{F}$ every graph with an \mathcal{F} -deletion set X has an (α, β) -protrusion decomposition where β is constant and $\alpha = O(|N[X]|)$.

Lemma 5 (Protrusion Decomposition Lemma). *If a n -vertex graph G has a vertex subset X such that $\mathbf{tw}(G \setminus X) \leq b$, then G admits a $((4|N[X]|)(b+1), 2(b+1))$ -protrusion decomposition. Furthermore, if we are given the set X then this protrusion decomposition can be computed in linear time. Here b is a constant.*

Proof. We give a proof for the case when X is explicitly given to us. The proof will automatically imply the existence of a $(4(b+1)|N(X)|, 2b+2)$ -protrusion decomposition of G for the case when we are just guaranteed the existence of X . The algorithm starts by computing a nice tree decomposition (T, \mathcal{B}) of $G \setminus X$ with width at most b . Notice that since b is a constant this can be done in linear time [6].

For every $v \in N(X)$ add a node u in T such that $v \in B_u$ to a set M' . We have that $M' \leq |N(X)|$. Let M' be the set of marked nodes and set $M = \mathbf{LCA-closure}(M')$. By Lemma 1, $M \leq 2|M'| \leq 2|N(X)|$. Let $Q_1, Q_2 \dots Q_t$ be the connected components of $T \setminus Q$. Since T is a binary tree $T \setminus M$ has at most $2|M|+1$ connected components, so $t \leq 4|N(X)|+1$. By Lemma 1 we have that for every $i \leq t$, $|N_T(Q_i)| \leq 2$.

Define $R_0 = X \cup \bigcup_{u \in M} B_u$ and for each $1 \leq i \leq t$ set $R_i = \bigcup_{u \in Q_i} B_u \setminus R_0$. Since every vertex of $G \setminus X$ appears in a bag of the tree-decomposition, R_0, \dots, R_t forms a partition of $V(G)$. By construction we have that for every $i \geq 1$, $N(R_i) \subseteq R_0$ and $\mathbf{tw}(G[N[R_i]]) \leq b$. Furthermore, since $|N_T(Q_i)| \leq 2$ we have $|N(R_i)| \leq 2(b+1)$. Thus $R_0 \dots R_t$ form a (α, β) -protrusion decomposition of G where $\beta \leq 2(b+1)$ and $\alpha \leq \max(|R_0|, t) \leq (4|N[X]|)(b+1)$. It is easy to implement a procedure that computes $R_0 \dots R_t$ in this way in linear time. \square

We are now in a position to prove Lemma 3

Proof of Lemma 3. We need to prove that for every $\mathcal{F} \in \mathcal{F}$ there exist constants r and α such that if a graph G has no r -protrusion of size at least r' , then every minimal \mathcal{F} -deletion set S of G is a $\frac{\alpha}{r'}$ -cover of G . By Proposition 1 there exists a constant η depending only on \mathcal{F} such that $\mathbf{tw}(G \setminus S) \leq \eta$. By Lemma 5, G has a $((4|N[S]|)(\eta+1), 2(\eta+1))$ -protrusion decomposition $R_0 \dots R_t$. Set $r = 2(\eta+1)$ and suppose G has no r -protrusions of size at least r' . Then $t \leq (4|N[S]|)(\eta+1)$, $|R_0| \leq (4|N[S]|)(\eta+1)$ and so $|V(G)| = |R_0| + \sum_i |R_i| \leq (4|N[S]|)(\eta+1)(r'+1) \leq (8|N[S]|)(\eta+1)r'$. Since $\mathbf{tw}(G \setminus S) \leq \eta+1$ it follows that $G \setminus S$ is $(\eta+1)$ -degenerate and so $\sum_{v \in V(G) \setminus S} d(v) \leq (8|N[S]|)(\eta+1)^2 r'$. Set $\alpha = \frac{1}{18(\eta+1)^2}$ and observe that

$$\sum_{v \in V(G)} d(v) \leq \sum_{v \in S} d(v) + \sum_{v \in V(G) \setminus S} d(v) \leq \sum_{v \in S} d(v) + (8|N[S]|)(\eta+1)^2 r' \leq \frac{r}{\alpha} \cdot \sum_{v \in S} d(v).$$

The last inequality follows from the fact that there are no isolated vertex in S . \square

4 Fast Protrusion Replacement

What makes the polynomial factor of Algorithm 2 large is the algorithm of Lemma 2 to remove all large enough protrusions with small border size. In this section we give much faster algorithms that reduce “almost all” large protrusions with small border. We then

show that reducing almost all protrusions instead of all protrusions is sufficient to obtain the conclusion of Lemma 3. The “fast protrusion reduction” algorithms we design in this section are applicable to any problem that uses protrusion reducer, and hence they are useful well beyond the scope of this paper. We give two algorithms for fast protrusion replacement, a randomized algorithm and a slightly slower deterministic algorithm.

The Randomized Fast Protrusion Replacer. We now describe an algorithm that we call the *Randomized Fast Protrusion Replacer (RFPR)*. The algorithm works for parameterized graph problems Π that have a protrusion replacer, takes as input an instance (G, k) and outputs another instance (G', k') . Just as a normal protrusion replacer, the RFPR is actually a family of algorithms with one algorithm for each value of the integer r . We describe how the algorithm proceeds for a fixed value of r . Let r' be the smallest integer such that the protrusion replacer for Π replaces r -protrusions of size at least r' .

The RFPR proceeds as follows. We select a random partition of $V(G)$ into $r + 1$ sets X_1, X_2, \dots, X_{r+1} . For every $i \leq r + 1$ we compute the connected components of $G[X_i]$ and add these components to a collection \mathcal{C}' . This results in a partition of $V(G)$ into $\mathcal{C}' = C'_1, C'_2, \dots, C'_{t'}$. Now, discard every component C'_i such that $N(C'_i) > r$ and every component C_i such that $\text{tw}(G[N[C_i]]) > r$. Discarding all of these components can be done in linear time - the only computationally hard step is to check whether the treewidth of the components is at most r , this can be done in linear time using Bodlaenders’s algorithm [6]. Let $\mathcal{C}^* = C_1^*, \dots, C_{t^*}^*$ be the remaining components.

For every $C_i^* \in \mathcal{C}^*$, $N[C_i^*]$ is a r -protrusion in G . However some, if not all of the components in \mathcal{C}^* could have less than r' vertices and so the protrusion replacer can’t reduce them. However it could be possible to group some components in \mathcal{C}^* with the same neighbourhood together such that their union is a protrusion that is large enough to be reduced. From \mathcal{C}^* we will compute a collection \mathcal{R} of disjoint vertex sets such that for every $R \in \mathcal{R}$, $N[R]$ is an r -protrusion in G of size at least r' . Our aim is to compute such a set with $|\mathcal{R}|$ being large. For every component $C_i^* \in \mathcal{C}^*$ of size at least r' we add C_i^* to \mathcal{R} and remove C_i^* from \mathcal{C}^* . Let $\mathcal{C} = C_1 \dots C_t$ be the remaining components. All components in \mathcal{C} have size at most r' . Set R_{big} to be the number of components $C_i^* \in \mathcal{C}^*$ on at least r' vertices that are added to \mathcal{R} .

Now we partition \mathcal{C} into groups according to the neighbourhood of the components. Specifically we compute a partition of \mathcal{C} into $\mathcal{Z}_1, \dots, \mathcal{Z}_q$ such that for every pair $C_i \in \mathcal{C}$, $C_{i'} \in \mathcal{C}$ such that $N(C_i) = N(C_{i'})$, C_i and $C_{i'}$ are in the same \mathcal{Z}_j , while for every pair $C_i \in \mathcal{C}$, $C_{i'} \in \mathcal{C}$ such that $N(C_i) \neq N(C_{i'})$ we have $C_i \in \mathcal{Z}_j \rightarrow C_{i'} \notin \mathcal{Z}_j$. Such a partition can be computed in time $O(nr)$ because every component in \mathcal{C} has at most r neighbours; First we sort the neighbor lists of each component according to some ordering of the vertex set, for example an arbitrary labelling of the vertices from 1 to n . Then we do r stable bucket sorts on \mathcal{C} sorting the components first on their first neighbour, then their second neighbor, etc.

Having computed the partitioning $\mathcal{Z}_1, \dots, \mathcal{Z}_q$ we now compute \mathcal{R} as follows. As long as there is a \mathcal{Z}_i such that $\sum_{C_j \in \mathcal{Z}_i} |C_j| \geq r'$ select a minimal collection $\mathcal{Z} \subseteq \mathcal{Z}_i$ such that $\sum_{C_j \in \mathcal{Z}} |C_j| \geq r'$. Add $\bigcup_{C_j \in \mathcal{Z}} C_j$ to \mathcal{R} and remove the components of \mathcal{Z} from \mathcal{Z}_i . This procedure can easily be implemented in linear time. This concludes the construction of \mathcal{R} .

Given \mathcal{R} we proceed as follows, for a set $R \in \mathcal{R}$ we run the protrusion replacer for Π on (G, k) with protrusion $N[R]$. The protrusion replacer outputs an equivalent instance (G^*, k^*) with $|V(G^*)| < |V(G)|$. Here G^* is a graph where R has been replaced by a smaller protrusion R' . Since all the sets in \mathcal{R} are disjoint, the other sets in \mathcal{R} are now r -protrusions in G^* of size at least r' . Thus we can run the protrusion replacer on all the sets in \mathcal{R} . This takes time $\sum_{R \in \mathcal{R}} O(|R|) = O(n)$. Let (G', k') be the instance obtained after running the

protrusion replacer on all the sets in \mathcal{R} . The RFPR outputs the instance (G', k') . We collect a few simple facts about the RFPR in the following lemma.

Lemma 6. *Given an instance (G, k) , the RFPR runs in time $O(n+m)$, computes a collection \mathcal{R} of protrusions and outputs an equivalent instance (G', k') , such that $|V(G')| \leq |V(G)| - |\mathcal{R}|$. Furthermore $\mathcal{R} \geq R_{big} + \left\lceil \sum_{i \leq q} \frac{\sum_{C \in \mathcal{Z}_i} |C| - (r'-1)}{2(r'-1)} \right\rceil$.*

Proof. The instances (G, k) and (G', k') are equivalent because (G', k') is obtained from (G, k) by repetitive applications of a protrusion replacer. In the description of the algorithm we made sure that each individual stage of the algorithm runs in linear time. Finally, each application of the protrusion replacer reduces the size of the graph by at least one. We apply the protrusion replacer $|\mathcal{R}|$ times. Hence $|V(G')| \leq |V(G)| - |\mathcal{R}|$.

Finally, when the RFPR selects a minimal collection $\mathcal{Z} \subseteq \mathcal{Z}_i$ such that $\sum_{C_j \in \mathcal{Z}} |C_j| \geq r'$, since each $C_j \in \mathcal{Z}_i$ has size at most r' it follows that $\sum_{C_j \in \mathcal{Z}} |C_j| \leq 2(r'-1)$. Thus every time we add a set to \mathcal{R} , $\sum_{C \in \mathcal{Z}_i} |C|$ decreases by at most $2(r'-1)$. At the end when we can not add more sets to \mathcal{R} we have that for every i , $\sum_{C \in \mathcal{Z}_i} |C| \leq r'$. This proves the last part of the statement of the lemma. \square

Analyzing the Randomized Fast Protrusion Replacer. We now analyze how many vertices the Fast Protrusion Replacer reduces the instance by. To that end we need to define the notion protrusion covers.

Definition 2. *An (a, b, r) -protrusion cover in a graph G is a collection $\mathcal{Z} = \mathcal{Z}_1, \dots, \mathcal{Z}_t$ of sets such that for every i , $N[\mathcal{Z}_i]$ is a r -protrusion in G and $a \leq |\mathcal{Z}_i| \leq b$, and for every $i \neq j$, $\mathcal{Z}_i \cap \mathcal{Z}_j = \emptyset$ and there are no edges from \mathcal{Z}_i to \mathcal{Z}_j . The size of \mathcal{Z} is $|\mathcal{Z}|$.*

Lemma 7. *Let Π be a problem that has a protrusion replacer which replaces r -protrusions of size at least r' , and let $s \geq r' \cdot 2^r$. If G is a graph with a $(s, 6s, r)$ -protrusion cover \mathcal{X} , then if the RFPR is run on (G, k) , with probability at least $1 - e^{-\frac{|\mathcal{X}|}{8(r+1)^{6s}}}$ the output instance (G', k') satisfies $|V(G)| - |V(G')| \geq \frac{|\mathcal{X}|}{4(r+1)^{6s}}$.*

Proof. By Lemma 6 the RFPR computes a set \mathcal{R} of protrusions and $|V(G)| - |V(G')| \geq |\mathcal{R}|$. Thus it is sufficient to show that with high probability, $\mathcal{R} \geq \frac{|\mathcal{X}|}{4(r+1)^{6s}}$. Define $\bar{X} = V(G) \setminus \bigcup_{X \in \mathcal{X}} X$. Since no edge goes between different sets in \mathcal{X} we have that for every $X \in \mathcal{X}$, $N(X) \subseteq \bar{X}$. The only randomized step of the RFPR is the initial partitioning of $V(G)$ into sets X_1, \dots, X_{r+1} . We may think of this partitioning step as selecting a random coloring of $V(G)$ with colors from $\{1, \dots, r+1\}$.

We say that a set X in \mathcal{X} *succeeds* if all vertices in X are colored with the same color, and no vertex of $N(X)$ is colored with that color. Since every set $X \in \mathcal{X}$ has at most r neighbours we have that the probability that X succeeds given any coloring of \bar{X} is at least $\frac{1}{(r+1)^{|\bar{X}|}}$. Hence the expected number of sets $X \in \mathcal{X}$ that succeed is at least $\frac{|\mathcal{X}|}{(r+1)^{6s}}$. Suppose t sets succeed. We prove that the set \mathcal{R} constructed by the Randomized Fast Protrusion Replacer has size at least $t/2$.

For each set X that succeeds, the connected components of X are added to \mathcal{C}' , and since they all have treewidth at most r and have at most r neighbors, none of them are discarded. Hence the connected components of X are all added to \mathcal{C}^* . Since $|Z| \geq r' \cdot 2^r$, if we group the connected components of Z by their neighbourhood, at least one group has combined size at least r' . If this group contains a connected component on at least r' vertices then this component is added to \mathcal{R} directly and X contributes one to R_{big} . If this group does

not contain any components of size at least r' then the group is added in its entirety to some set \mathcal{Z}_i . In this case the group contributes at least r' to $\sum_{C \in \mathcal{Z}_i} |C|$. By Lemma 6, $\mathcal{R} \geq R_{big} + \left\lceil \sum_{i \leq q} \frac{\sum_{C \in \mathcal{Z}_i} |C| - (r' - 1)}{2(r' - 1)} \right\rceil$. Hence the total number of sets added to \mathcal{R} is at least $t/2$.

Since the neighbourhoods of different sets in \mathcal{X} may overlap there are dependencies between which sets succeed. However, given any coloring of \bar{X} the success of different sets in \mathcal{X} is independent, since whether X succeeds or not depends only on the color of vertices in X and the color of vertices in $N(X) \subseteq \bar{X}$. Thus for every coloring of \bar{X} the number of sets that succeed is a sum of independent 0-1 variables taking value 1 with probability at least $\frac{1}{(r+1)^{|\bar{X}|}}$. Standard Chernoff bounds for the binomial distribution show that if T is a sum of n independent 0-1 variables taking value 1 with probability p , then $P[X \leq np/2] < e^{-\frac{np}{8}}$. Plugging this in for the number of sets in \mathcal{X} that succeed yields that the probability that $|\mathcal{R}| \leq \frac{|\mathcal{X}|}{4(r+1)^{6s}}$ is at most $e^{-\frac{|\mathcal{X}|}{8(r+1)^{6s}}}$. \square

The Deterministic Fast Protrusion Replacer We prove that the RFPR can be made deterministic at the cost of a $\log n$ factor in the running time. The only randomized step of the RFPR is the initial step where the vertices of G are partitioned into $r+1$ sets X_1, \dots, X_{r+1} . We may think of this partitioning step as selecting a random coloring of $V(G)$ with colors from $\{1, \dots, r+1\}$. The main difference between the randomized and the deterministic Fast Protrusion Replacer is how this coloring is chosen. The Deterministic Fast Protrusion Replacer only partitions $V(G)$ in two sets X_1 and X_2 - this corresponds to coloring the vertices with colors 1 and 2. To describe the colorings the Deterministic Fast Protrusion Replacer (DFPR) uses we use the notion of *universal sets*.

Definition 2 ([35]). A (n, t) -universal set \mathcal{P} of a ground set U on n elements is a collection \mathcal{P} of subsets of U such that for every set $S \subseteq U$ and set $S' \subseteq S$ there is a set $P \in \mathcal{P}$ such that $P \cap S = S'$.

Theorem 4 ([35]). There is a deterministic algorithm with running time $O(2^{t+o(t)} n \log n)$ that constructs an (n, t) -universal set \mathcal{P} such that $|\mathcal{P}| = 2^{t+o(t)} \log n$.

The DFPR has two parameters, r and s , instead of just one parameter r . It constructs a $(n, 6s+r)$ -universal set \mathcal{P} in time $O(2^{6s+r+o(6s+r)} n \log n) = O(2^{20s} n \log n)$ and selects the first set $P \in \mathcal{P}$. It sets $X_1 = P$, $X_2 = V(G) \setminus P$ and then it proceeds just as the RFPR would. For a fixed set $P \in \mathcal{P}$ this takes linear time and will reduce (G, k) to an equivalent instance (G', k') . Choosing different sets $P \in \mathcal{P}$ results in different output instances (G', k') . The DFPR tries all possible choices for $P \in \mathcal{P}$ and then finally outputs the instance (G', k') that maximizes $|V(G)| - |V(G')|$. The total time taken by the DFPR is $O((2^{20s} n \log n) + |\mathcal{P}| \cdot O(n + m)) = O((2^{20s}(n + m) \log n))$. This proves the following lemma.

Lemma 8. Given an instance (G, k) , the DFPR runs in time $O((2^{20s}(n+m) \log n))$, computes a collection \mathcal{R} of protrusions and outputs an equivalent instance (G', k') , such that $|V(G')| \leq |V(G)| - |\mathcal{R}|$.

We now give a lemma analogous to Lemma 7 for the DFPR.

Lemma 9. Let Π be a problem that has a protrusion replacer which replaces r -protrusions of size at least r' , and let $s \geq r' \cdot 2^r$. If G is a graph with a $(s, 6s, r)$ -protrusion cover \mathcal{X} , then if the RFPR is run on (G, k) , the output instance (G', k') satisfies $|V(G)| - |V(G')| \geq \frac{|\mathcal{X}|}{2^{20s} \log n}$.

Proof. In the proof of Lemma 7 we showed that $|V(G)| - |V(G')|$ is lower bounded by the number of sets that succeeds. Since each set $X \in \mathcal{X}$ has size at most $6s$ and $|N[X]| \leq 6s + r \leq 7s$ it follows that for every $X \in \mathcal{X}$ there is some coloring set $P \in \mathcal{P}$ that makes X succeed. Hence there is a coloring $P \in \mathcal{P}$ that makes at least $\frac{|\mathcal{X}|}{|\mathcal{P}|} \geq \frac{|\mathcal{X}|}{2^{r+6s+o(r+6s)} \log n}$ sets succeed. In the proof of Lemma 7 we showed that $|V(G)| - |V(G')|$ is at least half the number of succeeding sets. Since $2 \cdot 2^{r+6s+o(r+6s)} \log n \leq 2^{20s}$ we have $|V(G)| - |V(G')| \geq \frac{|\mathcal{X}|}{2^{20s} \log n}$. \square

We now proceed to prove that if G has a protrusion decomposition such that a linear fraction of the vertices appear in large enough r -protrusions then with high probability the Randomized Fast Protrusion Replacer will reduce G by a linear fraction of its vertices. To that end we need to have a closer look at the relationship between protrusion decompositions and protrusion covers.

Protrusion Covers from Protrusion Decompositions. First we prove that in a graph of small treewidth we can always find protrusion covers with large size.

Lemma 10. *There exists a constant c such that for any integers $n \geq s > b \geq 2$ and n -vertex graph G of treewidth b , G has a $(s, 6s, 2(b+1))$ cover of size at least $\frac{n}{122s}$.*

Proof. Let (T, \mathcal{B}) be a nice tree-decomposition of G of width b . For a subset $Q \subseteq V(T)$ by $P(Q)$ we denote $\cup_{q \in Q} B_q$. For a rooted tree T , and a vertex $v \in T$, a component C of $T \setminus \{v\}$ is said to be below v if all vertices of C are descendants of v in T . We start by constructing a set $S \subseteq V(T)$ and a collection $Q_1, \dots, Q_{|S|}$ of connected components of $T \setminus S$ using the following greedy procedure.

Let r be the root of T . In the beginning $S = \emptyset$ and $T^r = T$. We maintain a loop invariant that T^r is the connected component of $T \setminus S$ that contains r . Now, at step i of the greedy procedure we pick a lowermost vertex v_i in $V(T^r)$ such that there is a connected component Q_i of $T^r \setminus \{v_i\}$ below v_i such that $|P(Q_i)| \geq 3s + 7(b+1)$. Now we add v_i to S and update T^r accordingly. The procedure terminates when no vertex v in T^r has this property. In particular, if for any $v \in T^r$, every component Q of $T^r \setminus \{v\}$ below v , $|P(Q)| < 3s + 7(b+1)$, the procedure terminates. Since (T, \mathcal{B}) is a nice tree decomposition, we have that for any vertex $v \in T_r$ and parent u of v , if C_v and C_u are the components of $T^r \setminus \{v\}$ and $T^r \setminus \{u\}$ maximizing $|P(C_v)|$ and $|P(C_u)|$ respectively, then $|P(C_u)| \leq 2|P(C_v)|$. Hence we know that for every component Q of $T \setminus S$, $|P(Q)| < 6s + 14(b+1) \leq 20s$. This bound holds both for the components included in the collection $Q_1, \dots, Q_{|S|}$ and the ones that do not.

Having constructed S and $Q_1, \dots, Q_{|S|}$ we let $S' = \mathbf{LCA-closure}(S)$. By Lemma 1 we have $|S'| \leq 2|S|$. Let $S^* = S' \setminus S$. Since $|S^*| \leq |S|$, at most $\frac{|S|}{2}$ of the components $Q_1, \dots, Q_{|S|}$ contain at least two vertices of S^* . This implies that at least $\frac{|S|}{2}$ of the components $Q_1, \dots, Q_{|S|}$ contain at most one vertex of S^* . Without loss of generality, let $Q_1, \dots, Q_{|S|/2}$ contain at most one vertex of S^* each. For every $i \leq |S|/2$, if Q_i contains no vertex of S^* then $Q'_i = Q_i$ is a component of $Q \setminus S'$ with $|P(Q'_i)| \geq 3s + 7(b+1) \geq s + 2(b+1)$. If Q_i contains one vertex v of S^* , since v has degree at most 3 and $|P(Q_i)| \geq 3s + b$, $Q_i \setminus \{v\}$ has at least one component Q'_i with $|P(Q'_i)| \geq s + 2(b+1)$. Thus we have constructed a set S' and a collection of components $Q'_1, \dots, Q'_{|S|/2}$ of $T \setminus S'$ of size at least $s + 2(b+1)$. By Lemma 1 every Q'_i has at most two neighbors in T .

We make a collection \mathcal{Z} as follows. For every $i \leq |S|/2$ let $Z_i = P(Q'_i) \setminus P(S')$. Since Q'_i has at most two neighbors in T it follows that $N[Z_i]$ is a $2(b+1)$ -protrusion and that $|Z_i| \geq s + 2(b+1) - 2(b+1) = s$. We have already shown that $|Q'_i| \leq 20s$ so $|Z_i| \leq 20s$ as well. Hence \mathcal{Z} is in fact a $(s, 6s, 2(b+1))$ -protrusion cover of G . It remains to lower bound

$|\mathcal{Z}|$. We have that $|\mathcal{Z}| = |S|/2$. Furthermore we have that S , together with the connected components of $T \setminus S$ cover T . Since every bag has size at most $(b+1) \leq s$, $T \setminus S$ has at most $2|S| + 1 \leq 3|S|$ connected components and for every component Q of $T \setminus S$, $|P(Q)| \leq 20s$ we have that $|S|(b+1) + 3|S| \cdot 20s \geq n$. Since $s \geq b+1$ this implies that $|S| \geq \frac{n}{122s}$. \square

Lemma 11. *If G has an (α, β) -protrusion decomposition, then for every $s > \beta$, G has a $(s, 6s, 3(\beta+1))$ -protrusion cover of size at least $\frac{n}{122s} - \alpha$.*

Proof. Let R_0, \dots, R_t be an (α, β) -protrusion decomposition of G . At most α vertices are in R_0 , and at most $\alpha \cdot s$ vertices are in sets R_i for $i \geq 1$ such that $|R_i| < s$. For each $i \geq 1$ such that $|R_i| \geq s$ we apply Lemma 10 and obtain a $(s, 6s, 2(\beta+1))$ -protrusion cover \mathcal{Z}_i in $G[R_i]$. We let \mathcal{Z} be the union of all the \mathcal{Z}_i 's constructed in this manner. For every $Z \in \mathcal{Z}_i$, $N_{G[R_i]}[Z_i]$ is a $2(\beta+1)$ -protrusion in $G[R_i]$. However Z might have neighbors also in R_0 . The number of neighbors of Z in R_0 is at most β and hence $N[Z]$ is a $3(\beta+1)$ -protrusion in G . We conclude that \mathcal{Z} is a $(s, 6s, 3(\beta+1))$ -protrusion cover in G . The size of \mathcal{Z} is at least $\frac{n - \alpha - \alpha \cdot s}{122s} \geq \frac{n}{122s} - \alpha$. \square

The Fast Protrusion Replacer Theorems We are now ready to prove our main results on Fast Protrusion Replacement.

Theorem 5 (Randomized Fast Protrusion Replacer Theorem). *Let Π be a problem that has a protrusion replacer that replaces r protrusions of size at least r' , and let s and β be constants such that $r \geq 3(\beta+1)$ and $s \geq 2^r \cdot r'$. Given an instance (G, k) as input, the RFPR will run in time $O(n+m)$ and produce an equivalent instance (G', k') with $|V(G')| \leq |V(G)|$ and $k' \leq k$. If additionally G has a (α, β) -protrusion decomposition such that $\alpha \leq \frac{n}{244s}$, then with probability at least $1 - e^{-\frac{n}{2000s(r+1)^{6s}}}$ we have $|V(G)| - |V(G')| \geq \frac{n}{1000(r+1)^{6s}}$.*

Proof. The first part of the statement follows directly from Lemma 6. If G has a (α, β) -protrusion decomposition such that $\alpha \leq \frac{n}{244s}$, then by Lemma 11, G has a $(s, 6s, 3(\beta+1))$ -protrusion cover \mathcal{X} of size at least $\frac{n}{122s} - \alpha \geq \frac{n}{244s}$. Plugging \mathcal{X} into Lemma 7 yields that with probability at least $1 - e^{-\frac{|\mathcal{X}|}{8s(r+1)^{6s}}} \geq 1 - e^{-\frac{n}{2000s(r+1)^{6s}}}$ we have $|V(G)| - |V(G')| \geq \frac{|\mathcal{X}|}{4(r+1)^{6s}} \geq \frac{n}{1000(r+1)^{6s}}$. \square

Theorem 6 (Deterministic Fast Protrusion Replacer Theorem). *Let Π be a problem that has a protrusion replacer that replaces r protrusions of size at least r' , and let s and β be constants such that $r \geq 3(\beta+1)$ and $s \geq 2^r \cdot r'$. Given an instance (G, k) as input, the DFPR will run in time $O(2^{20s} \cdot (n+m) \log n)$ and produce an equivalent instance (G', k') with $|V(G')| \leq |V(G)|$ and $k' \leq k$. If additionally G has a (α, β) -protrusion decomposition such that $\alpha \leq \frac{n}{244s}$ then we have $|V(G)| - |V(G')| \geq \frac{n}{244 \cdot 2^{20s} \log n}$.*

Proof. The first part of the statement follows directly from Lemma 8. If G has a (α, β) -protrusion decomposition such that $\alpha \leq \frac{n}{244s}$, then by Lemma 11, G has a $(s, 6s, 3(\beta+1))$ -protrusion cover \mathcal{X} of size at least $\frac{n}{122s} - \alpha \geq \frac{n}{244s}$. Plugging \mathcal{X} into Lemma 9 yields that $|V(G)| - |V(G')| \geq \frac{|\mathcal{X}|}{2^{20s} \log n} \geq \frac{n}{244 \cdot 2^{20s} \log n}$. \square

It can be shown that Theorem 5 could replace the simple protrusion reduction algorithm of Lemma 2 and make thus Algorithm 2 run in linear time. However we are first going to refine Algorithm 2 even further so that it becomes simultaneously single exponential parameterized algorithm and an approximation algorithm for \mathcal{F} -DELETION for all connected $\mathcal{F} \in \mathcal{F}$. To that end we develop the notion of lossless protrusion replacement.

5 Lossless Protrusion Replacement

In this section we develop the notion of lossless protrusion replacement. We consider *CMSO* vertex subset problems. In a MIN-CMSO vertex subset problem, Π , we are given a graph G as input. The objective is to find a set $S \subseteq V(G)$ minimizing $|S|$ such that the CMSO-expressible predicate $P_\Pi(G, S)$ is satisfied. Similarly, in a MAX-CMSO vertex subset problem, Π , we are given a graph G as input. The objective is to find a set $S \subseteq V(G)$ maximizing $|S|$ such that the CMSO-expressible predicate $P_\Pi(G, S)$ is satisfied. Given a MIN-CMSO (MAX-CMSO) vertex subset problem, Π and an input graph G to Π , by $OPT(G)$ we denote the size of the smallest (largest) set S such that the CMSO-expressible predicate $P_\Pi(G, S)$ is satisfied. Next we define the notion of a lossless protrusion replacer. A lossless protrusion replacer is essentially a protrusion replacer that reduces protrusions in such a way that any feasible solution to the reduced instance can be changed into a feasible solution of the original instance without changing the gap between the feasible solution and the optimum. The notion of lossless protrusion replacement is central in our approximation algorithms.

Definition 3 (Lossless Protrusion Replacer). *A lossless protrusion replacer for MIN-CMSO (MAX-CMSO) vertex subset problem Π is a family of algorithms, with one algorithm for every constant r . The r 'th algorithm has the following specifications. There exists a constant r' (which depends on r) such that given an instance G and an r -protrusion X in G of size at least r' , the algorithm runs in time $O(|X|)$ and outputs an instance G' with the following properties.*

- G' is obtained from G by replacing X by a r -boundaried graph X' with less than r' vertices and thus $|V(G')| < |V(G)|$.
- $OPT(G') \leq OPT(G)$.
- There is an algorithm that runs in $O(|X|)$ time and given a feasible solution S' to G' outputs a set $X^* \subseteq X$ such that $S = (S' \setminus X') \cup X^*$ is a feasible solution to G and $|S| \leq |S'| + OPT(G) - OPT(G')$.

We would like to give sufficient conditions for a problem to have a lossless protrusion replacer. An ideal setting would be that every graph optimization problem that has finite integer index when parameterized by the size of the optimal solution has a lossless protrusion replacer. Unfortunately such a theorem seems to be out of reach, and it is quite possible that this is not true. However, in [8] a sufficient condition is given for a *CMSO* vertex subset problem to have finite integer index. This condition is called *strong monotonicity* and it is proved that every *CMSO* vertex subset problem that is strongly monotone has finite integer index and hence has a protrusion replacer. It turns out that strong monotonicity is a sufficient condition for a *CMSO* vertex subset problem to not only have a protrusion replacer, but also a lossless protrusion replacer. We now prove this fact.

Let Π be a MIN-CMSO problem and \mathcal{F}_t be the set of pairs (G, S) where G is a t -boundaried graph and $S \subseteq V(G)$. For a t -boundaried graph G we define the function $\zeta_G : \mathcal{F}_t \rightarrow \mathbb{N} \cup \{\infty\}$ as follows. For a pair $(G', S') \in \mathcal{F}_t$, if there is no set $S \subseteq V(G)$ such that $P_\Pi(G \oplus G', S \cup S')$ holds, then $\zeta_G((G', S')) = \infty$. Otherwise $\zeta_G((G', S'))$ is the size of the smallest $S \subseteq V(G)$ such that $P_\Pi(G \oplus G', S \cup S')$ holds. If Π is a MAX-CMSO problem then we define $\zeta_G((G', S'))$ to be the size of the largest $S \subseteq V(G)$ such that $P_\Pi(G \oplus G', S \cup S')$ holds. If there is no set $S \subseteq V(G)$ such that $P_\Pi(G \oplus G', S \cup S')$ holds, then $\zeta_G((G', S')) = \infty$.

Definition 3 ([8]). *A MIN-CMSO problem Π is said to be strongly monotone if there exists a function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that the following condition is satisfied. For every t -boundaried*

graph G , there is a subset $S \subseteq V(G)$ such that for every $(G', S') \in \mathcal{F}_t$ such that $\zeta_G((G', S'))$ is finite, $P_\Pi(G \oplus G', S \cup S')$ holds and $|S| \leq \zeta_G((G', S')) + f(t)$.

Definition 4 ([8]). A MAX-CMSO problem Π is said to be strongly monotone if there exists a function $f : \mathbb{N} \rightarrow \mathbb{N}$ such that the following condition is satisfied. For every t -boundaried graph G , there is a subset $S \subseteq V(G)$ such that for every $(G', S') \in \mathcal{F}_t$ such that $\zeta_G((G', S'))$ is finite, $P_\Pi(G \oplus G', S \cup S')$ holds and $|S| \geq \zeta_G((G', S')) - f(t)$.

Theorem 7. Every MIN-CMSO or MAX-CMSO vertex subset problem Π , that is also strongly monotone admits a lossless protrusion replacer.

Before proving the theorem we will need an auxiliary lemma.

Lemma 12. If a graph G contains an r -protrusion X where $|X| > c > 0$, then it also contains a $(2r + 1)$ -protrusion Y where $c < |Y| \leq 2c$. Moreover, given X we can compute Y and a tree decomposition of Y of width $\leq 2r$ in $O(|X|)$ time.

Proof. Let (T, \mathcal{X}) be a nice tree decomposition of $G[X]$ rooted at a node r . We can compute (T, \mathcal{X}) from $G[X]$ in time $O(|X|)$ using Bodlaender's algorithm [6]. If $|X| \leq 2c$, we are done. Given a vertex x of the rooted tree T , we denote by $\mathcal{D}_T(x)$ the subset of $V(T)$ containing x and all its descendants in T . Let B_T be the set containing each vertex x of T with the property that the vertices appearing in $\bigcup_{y \in \mathcal{D}_T(x)} X_y$ (i.e. the vertices of the nodes corresponding to x and its descendants) are more than c . As $|X| \geq 2c$, B_T is a non-empty set. We choose b to be a member of B_T whose descendants do not belong in B_T . This choice of b ensures that $c < |\bigcup_{y \in \mathcal{D}_T(b)} X_y| \leq 2c$. We define $Y = \partial_G X \cup \bigcup_{y \in \mathcal{D}_T(b)} X_y$. As $G[Y]$ is an induced subgraph of X it follows that $\text{tw}(G[Y]) \leq r$. Furthermore $\partial_G(Y) \subseteq \partial_G X \cup X_b$, therefore Y is a $(2r + 1)$ -protrusion of G . \square

Proof of Theorem 7. We prove the theorem for MIN-CMSO problems; the proof for MAX-CMSO problems is similar. Let Π be a monotone MIN-CMSO problem. We define a partial order \leq_Π on pairs (G, S) such that G is a t -boundaried graph and $S \subseteq V(G)$. We say that $(G, S) \leq_\Pi (G', S')$ if for every (G_3, S_3) , $P_\Pi(G \oplus G_3, S \cup S_3) \rightarrow P_\Pi(G' \oplus G_3, S' \cup S_3)$. We say that $(G, S) \equiv_\Pi (G', S')$ if $(G, S) \leq_\Pi (G', S')$ and $(G', S') \leq_\Pi (G, S)$. Clearly \equiv_Π is an equivalence relation and since P_Π is a CMSO-expressible predicate it follows from [10, 16] that for every fixed t , \equiv_Π has finitely many equivalence classes. Thus there exists finite set \mathcal{S} of pairs (G_R, S_R) such that for every (G, S) there is a $(G_R, S_R) \in \mathcal{S}$ such that $(G, S) \equiv (G_R, S_R)$. We say that a pair (G, S) is *bad* if there is no (G', S') such that $P_\Pi(G \oplus G', S \cup S')$ holds. A pair that is not bad is called *useful*. Let \mathcal{U} be the set of all useful pairs in \mathcal{S} .

For a graph G and pair $(G_R, S_R) \in \mathcal{U}$ define $\gamma_G(G_R, S_R)$ to be the size of the smallest set $S \subseteq V(G)$ such that $(G_R, S_R) \leq_\Pi (G, S)$. If no such set S exists, $\gamma_G(G_R, S_R) = \infty$. We now prove that for any G , the maximum finite value of γ_G and the minimum (finite) value of γ_G differs by at most $f(t)$. Let $S \subseteq V(G)$ be the set such for every $(G', S') \in \mathcal{F}_t$ such that $\zeta_G((G', S'))$ is finite, $P_\Pi(G \oplus G', S \cup S')$ holds and $|S| \leq \zeta_G((G', S')) + f(t)$. Consider a useful pair $(G_R, S_R) \in \mathcal{U}$ such that $\gamma_G(G_R, S_R)$ is finite. Then there exists a set $S' \subseteq V(G)$ of size $\gamma_G(G_R, S_R)$ such that $(G_R, S_R) \leq_\Pi (G, S')$. Since $(G, S') \leq_\Pi (G, S)$ and S' is the smallest set such that $(G_R, S_R) \leq_\Pi (G, S')$ it follows that $|S'| \leq |S|$. On the other hand since (G_R, S_R) is useful there exists some (G^*, S^*) such that $P_\Pi(G_R \oplus G^*, S_R \cup S^*)$ holds. Then $P_\Pi(G \oplus G^*, S' \cup S^*)$ holds as well and hence $\zeta_G((G^*, S^*)) \leq |S'|$. Since $\zeta_G((G^*, S^*))$ is finite it follows that $|S| \leq \zeta_G((G^*, S^*)) + f(t) \leq |S'| + f(t)$. But this means that $|S| - f(t) \leq \gamma_G(G_R, S_R) \leq |S|$ and so the finite values of γ_G differ by at least $f(t)$. By the pigeon hole principle there exists a finite collection \mathcal{R} of t -boundaried graphs such that for any

t -boundaried G there is a $G_R \in \mathcal{R}$ and a constant $c_R \geq 0$ such that for every useful pair (G', S') , $\gamma_G(G', S') = \gamma_{G_R}(G', S') + c_R$. We call \mathcal{R} a *set of representatives* for (Π, t) .

For every integer c we define a relation \prec_c on t -boundaried graphs. We say that $G_1 \prec_c G_2$ if for every useful pair (G, S) , $\gamma_{G_1}(G, S) + c = \gamma_{G_2}(G, S)$. Observe that if $G_1 \prec_c G_2$ then $G_2 \prec_{-c} G_1$. Also, we have just shown that for every G there is a $G_R \in \mathcal{R}$ and constant $c_R \geq 0$ such that $G_R \prec_{c_R} G$. We now show that if $G \prec_c G'$ then for any t -boundaried graph G_3 and feasible solution S to Π on $G \oplus G_3$, there is a set $X^* \subseteq V(G')$ depending only on $S \cap V(G)$ and G such that $S' = X^* \cup S \setminus V(G)$ is also a feasible solution to Π on $G' \oplus G_3$ and $|S'| \leq |S| + c$.

Let $G \prec_c G'$ and consider a t -boundaried G_3 and a feasible solution S of Π on $G \oplus G_3$. Let $S_G = S \cap V(G)$ and $S_3 = S \setminus S_G$. (G, S_G) is a useful pair and so there is a pair $(G_R, S_R) \in \mathcal{U}$ such that $(G_R, S_R) \equiv_{\Pi} (G, S_G)$. Thus $\gamma_G(G_R, S_R) \leq |S_G|$ and hence $\gamma_{G'}(G_R, S_R) \leq |S_G| + c$. There is a set $X^* \subseteq V(G')$ such that $(G_R, S_R) \leq_{\Pi} (G', X^*)$ and $|X^*| \leq |S_G| + c$. The set X^* depends solely on (G_R, S_R) which depends solely on $S \cap V(G)$ and G . Furthermore, since $(G_R, S_R) \leq_{\Pi} (G', X^*)$ we have that $S' = X^* \cup S_3$ is also a feasible solution to Π on $G' \oplus G_3$ and $|S'| \leq |S_G| + c + |S_3| \leq |S| + c$.

We can now describe the lossless protrusion replacer for the problem Π . For parameter r consider the set \mathcal{R} of representatives for $(\Pi, 2(r+1))$. Let r' be the size of the largest graph in \mathcal{R} plus one. The lossless protrusion replacer for Π will reduce r -protrusions of size at least r' . Given an r -protrusion X of size at least r' we find a $2(r+1)$ protrusion $Y \subseteq X$ such that $r' \leq |Y| \leq 2r'$. This can be done in $O(|X|)$ time by Lemma 12. Consider now the $2(r+1)$ -boundaried graph $G_{Y \setminus \delta(Y)}^{\delta Y}$. There exists a $2(r+1)$ -boundaried graph $G_R \in \mathcal{R}$ and constant $c_R \geq 0$ such that $G_R \prec_{c_R} G_{Y \setminus \delta(Y)}^{\delta Y}$. Furthermore since $|Y| \geq r'$ we have that $|V(G_R)| < |Y|$. The protrusion replacer outputs the graph G' obtained by replacing Y by G_R in G .

For every subset $S_R \subseteq V(G_R)$ such that the pair (G_R, S_R) is useful, the protrusion replacer stores a subset $S_Y \subset Y$ such that $(G_R, S_R) \leq_{\Pi} (G_{Y \setminus \delta(Y)}^{\delta Y}, S_Y)$. Since $G_R \prec_{c_R} G_{Y \setminus \delta(Y)}^{\delta Y}$ there is such a set S_Y of size at most $|S_R| + c$. Now, for any feasible solution S in G' let $S_R = S \cup V(G_R)$. The pair (G_R, S_R) is useful and so the lossless protrusion replacer outputs the set $X^* = S_Y$ which it has stored for S_R . Now $S' = S_Y \cup (S \setminus V(G_R))$ is a feasible solution to G because $(G_R, S_R) \leq_{\Pi} (G_{Y \setminus \delta(Y)}^{\delta Y}, S_Y)$. Furthermore, since $|S_Y| \leq |S_R| + c$ we have that $|S'| \leq |S_Y| + |S \setminus V(G_R)| \leq |S_R| + c + |S \setminus V(G_R)| \leq |S| + c$. Thus it remains to prove that $c \leq OPT(G) - OPT(G')$, or in other words that $OPT(G') \leq OPT(G) - c$.

However $G_{Y \setminus \delta(Y)}^{\delta Y} \prec_{-c_R} G_R$, and hence for an optimal solution S of $G = G_{Y \setminus \delta(Y)}^{\delta Y} \oplus G_{V(G) \setminus Y}^{\delta}$ there is a feasible solution S' in $G_R \oplus G_{V(G) \setminus Y}^{\delta}$ of size at most $|S| - c_R$. Hence $OPT(G') \leq OPT(G) - c$ and the theorem follows. \square

Inserting a lossless protrusion replacer instead of a normal protrusion replacer into the Fast Protrusion Replacer algorithms directly yields the following theorems.

Theorem 8. *Let Π be a minimization (maximization) problem that has a lossless protrusion replacer that replaces r protrusions of size at least r' , and let s and β be constants such that $r \geq 3(\beta + 1)$ and $s \geq 2^r \cdot r'$. Given an instance G as input, the Randomized Fast Protrusion Replacer will run in time $O(n + m)$ and produce an instance G' with $|V(G')| \leq |V(G)|$. Given any feasible solution S' to G' a feasible solution S of G of size at most (at least) $|S'| - OPT(G') + OPT(G)$ can be computed in $O(n + m)$ time. If additionally G has a (α, β) -protrusion decomposition such that $\alpha \leq \frac{n}{244s}$, then with probability at least $1 - e^{-\frac{n}{2000s(r+1)^{6s}}}$ we have $|V(G)| - |V(G')| \geq \frac{n}{1000(r+1)^{6s}}$.*

Theorem 9. *Let Π be a minimization (maximization) problem that has a lossless protrusion replacer that replaces r protrusions of size at least r' , and let s and β be constants such that $r \geq 3(\beta + 1)$ and $s \geq 2^r \cdot r'$. Given an instance G as input, the Deterministic Fast Protrusion Replacer will run in time $O(2^{20s} \cdot (n + m) \log n)$ and produce an instance G' with $|V(G')| \leq |V(G)|$. Given any feasible solution S' to G' a feasible solution S of G of size at most (at least) $|S'| - \text{OPT}(G') + \text{OPT}(G)$ can be computed in $O(n + m)$ time. If additionally G has a (α, β) -protrusion decomposition such that $\alpha \leq \frac{n}{244s}$ then we have $|V(G)| - |V(G')| \geq \frac{n}{244 \cdot 2^{20s} \log n}$.*

6 Approximation and Fast Parameterized Algorithm for p - \mathcal{F} -DELETION

We are now ready to give the linear time, lossless variant of Lemma 2. Throughout this section $\text{OPT}(G)$ is the size of the smallest \mathcal{F} -deletion set of G , for the set \mathcal{F} currently under consideration. First we give an auxiliary lemma analyzing an execution of the Lossless RFPR on a graph with an \mathcal{F} -deletion set S .

Lemma 13. *For every connected $\mathcal{F} \in \mathcal{F}$, there exist constants $\rho, r, s, c < 1$ and $\gamma > 0$ such that if we run the Lossless RFPR with parameters r, s on a graph G which has a \mathcal{F} deletion set S which is not a ρ -cover, then with probability at least $1 - e^{-\gamma n}$ the output instance G' satisfies $V(G') \leq |V(G)|(1 - c)$.*

Proof. If G has a \mathcal{F} -deletion set S' which is not an ρ -cover, it also has a inclusion minimal \mathcal{F} -deletion set S which is not an ρ -cover. Such a minimal S contains no isolated vertices and hence satisfies $N[S] \leq 2 \sum_{v \in S} d(v) \leq 2\rho m$.

By Proposition 1, there exists a constant b such that $\text{tw}(G \setminus S) \leq b$. By Lemma 5, G has a $(4(b + 1)|N[S]|, 2(b + 1))$ -protrusion decomposition. Set $\beta = 2(b + 1)$, $r = 3(\beta + 1)$ and r' to be the smallest integer such that the lossless protrusion replacer will replace r -protrusions of size at least r' . Set $s = 2^r \cdot r'$. The protrusion decomposition of G is a $(4(b + 1)|N[S]|, \beta)$ -protrusion decomposition. By Theorem 8 there exist constants $0 < c < 1$ and $0 < \gamma$ such that if we run the Lossless RFPR on G and $4(b + 1)|N[S]| \leq \frac{n}{244s}$ then with probability at least $1 - e^{-\gamma n}$, the output graph G' satisfies $|V(G)| - |V(G')| \geq c|V(G)|$. We show that there is a constant $\rho < \frac{1}{3}$ such that if S is not a ρ -cover, then $|N[S]| \leq \frac{n}{1000(b+1)s}$.

Since $\text{tw}(G \setminus S) \leq b$ we have that $G \setminus S$ is $(b + 1)$ -degenerate. If S is not a ρ -cover then $m \leq n(b + 1) + \sum_{v \in S} d(v) \leq n(b + 1) + 2\rho m$. Rearranging yields that $N[S] \leq 2\rho m \leq n \frac{2\rho(b+1)}{1-2\rho} \leq n\rho 6(b + 1)$. Choosing $\rho = 6000(b + 1)^2 s$ yields that $|N[S]| \leq \frac{n}{1000(b+1)s}$. Hence, if S is not a ρ -cover then with probability at least $1 - e^{-\gamma n}$ the output instance G' of the Lossless RFPR satisfies $V(G') \leq |V(G)|(1 - c)$. \square

Lemma 14. *For every connected $\mathcal{F} \in \mathcal{F}$ there is an algorithm that given a graph G , takes $O(n + m)$ time and outputs a graph G' such that $V(G') \leq V(G)$ and $\text{OPT}(G') \leq \text{OPT}(G)$. Given a \mathcal{F} -deletion set S' of G' the algorithm can compute an \mathcal{F} -deletion set S of G of size $|S'| + \text{OPT}(G) - \text{OPT}(G')$ in time $O(n + m)$. Furthermore there exist a constant $0 < \rho < 1$ such that with probability at least $\frac{1}{2}$, every \mathcal{F} -deletion set S' of G' is a ρ -cover of G .*

Proof. By Lemma 13 there exist constants $\rho, r, s, c < 1$ and $\gamma > 0$ such that if we run the Lossless RFPR with parameters r, s on a graph G which has a \mathcal{F} deletion S which is not a ρ -cover, then with probability at least $1 - e^{-\gamma n}$ the output instance G' satisfies $V(G') \leq |V(G)|(1 - c)$. We set these constants as guaranteed by Lemma 13.

The algorithm sets $G_1 := G$, $i = 1$ and enters a loop that proceeds as follows. The algorithm runs the Lossless RFPR on G_i with parameters r and s , let the output of the Lossless RFPR be G_{i+1} . If $|V(G_{i+1})| > |V(G_i)|(1 - c)$ the algorithm halts and outputs G_i . Otherwise, the algorithm increments i and returns to the beginning of the loop.

The total time spent by the algorithm is upper bounded by a geometric series, and so the running time of the algorithm is $O(n + m)$. Similarly, by repeatedly applying Theorem 8 we can in linear time transform any \mathcal{F} -deletion set S_i of G_i back into a \mathcal{F} -deletion set S of G of size at most $|S'| + OPT(G) - OPT(G')$. It remains to prove that when the algorithm terminates, with probability at least $\frac{1}{2}$ we have that every \mathcal{F} -deletion set S' of G' is an ρ -cover of G .

The algorithm makes $t = O(\log n)$ calls to the Lossless RFPR. For $i \leq t + 1$ let $n_i = |V(G_i)|$. In call i , by Lemma 13, if G_i has an \mathcal{F} -deletion set S which is not a ρ -cover then the probability that $|V(G_{i+1})| > |V(G_i)|(1 - c)$ is at most $e^{-\gamma n_i}$. By the union bound the probability that this occurs at some step i is $\sum_{i \leq t} e^{-\gamma n_i}$. The n_i 's are a decreasing geometric series and so for a sufficiently large (constant) N we have that if $n_t \geq N$ then $\sum_{i \leq t} e^{-\gamma n_i} \leq 2e^{-\gamma n_t} \leq 1/2$.

Finally, if $n_t \leq N$ then any non-empty set S is a $\frac{1}{N^2}$ cover, and so if $\rho > \frac{1}{N^2}$ we can adjust ρ to $\frac{1}{N^2}$. This proves the lemma. \square

We are now ready to give the algorithm which is the main engine behind both our $2^{O(k)}n$ time algorithm and the quadratic approximation algorithm for \mathcal{F} -DELETION for connected sets $\mathcal{F} \in \mathcal{F}$.

Randomized- \mathcal{F} -Deletion(G)
Set $G_1 := G$ and $i := 1$
While (G_i is not \mathcal{F} -free) do as follows:

1. Apply Lemma 14 on G_i and obtain a new graph G'_i
2. Pick a vertex $u_i \in V(G'_i)$ at random with probability $\frac{d_{G'_i}(u)}{2|E(G'_i)|}$. Set $G_{i+1} := G'_i \setminus \{u_i\}$.
3. Increment i by 1.

Set $S_i = \emptyset$
For $j = i$ downto 2:

1. Set $S'_{j-1} := S_j \cup \{u_{j-1}\}$.
2. Apply Lemma 14 on G'_j and S'_j and obtain a set S_j .

Output $S := S_1$.

Figure 3: Randomized Algorithm for \mathcal{F} -DELETION for connected $\mathcal{F} \in \mathcal{F}$

We say that a *round* of Algorithm 3 is an iteration of the while-loop. Round x is the iteration when the value of i is x . The algorithm *succeeds* in round i if $OPT(G'_i) = OPT(G_{i+1}) + 1$ and it *fails* in round i otherwise. The *number of rounds* of a run of Algorithm 3 is the maximum value i takes. We make a series of observations about Algorithm 3. For every i we have that $|V(G'_i)| \leq |V(G_i)|$ and $|V(G_{i+1})| < |V(G'_i)|$. Hence we make the following observation.

Observation 1. *Algorithm 3 terminates after at most n rounds.*

The next observation follows directly from Lemma 14.

Observation 2. *The time taken in each round and each iteration of the for loop is $O(n+m)$.*

Next we prove that the algorithm always outputs feasible solutions.

Observation 3. *Algorithm 3 outputs an \mathcal{F} -deletion set of G .*

Proof. Let t be number of rounds. We have that G_t is \mathcal{F} -free and so $S_t = \emptyset$ is a \mathcal{F} -deletion set of G_t . If S_j is a \mathcal{F} -deletion set of G_j then $S'_j = S_j \cup \{u_{j-1}\}$ is a \mathcal{F} -deletion set of G'_{j-1} . Then, by Lemma 14, S_{j-1} is a \mathcal{F} -deletion set of G_{j-1} . Hence, by downward induction on j , S_1 is a \mathcal{F} -deletion set of $G_1 = G$. \square

Next we upper bound the size of the output solution S .

Lemma 15. *Let p be the number of rounds in which Algorithm 3 fails. Then the size of the output solution S is $|S| = OPT(G) + p$.*

Proof. For every x , define f_x to be the number of rounds $i \geq x$ such that the algorithm fails in round i . Let t be the number of rounds. We prove by downward induction on i that $|S_i| = OPT(G_i) + f_i$. Since $|S_t| = |f_t| = |OPT(G_t)| = 0$ this clearly holds for t . Consider now some $i < t$ such that the equation holds for $i + 1$.

If the algorithm succeeded in round i we have that $|S'_i| = |S_{i+1}| + 1$, that $OPT(G'_i) = OPT(G_{i+1}) + 1$ and that $f_i = f_{i+1}$ hence $|S'_i| = |S_{i+1}| + 1 = OPT(G_{i+1}) + f_{i+1} + 1 = OPT(G'_i) + f_i$. On the other hand if the algorithm fails in round i we have $|S'_i| = |S_{i+1}| + 1$, that $OPT(G'_i) = OPT(G_{i+1})$ and that $f_i = f_{i+1} + 1$. Then $|S'_i| = |S_{i+1}| + 1 = OPT(G_{i+1}) + f_{i+1} + 1 = OPT(G'_i) + f_i$. Hence in both cases we have that $|S'_i| = OPT(G'_i) + f_i$. By Lemma 14 we have that $|S_i| = |S'_i| + OPT(G_i) - OPT(G'_i) = OPT(G_i) + f_i$. This concludes the proof. \square

Now we lower bound the success probability of any round i of Algorithm 3.

Lemma 16. *There is a constant $p > 0$ such that the probability that Algorithm 3 succeeds in any given round i is at least p .*

Proof. By Lemma 14 there is a constant ρ such that with probability $1/2$, every \mathcal{F} -deletion set of G'_i is a ρ -cover. Let S^* be an optimal \mathcal{F} -deletion set of G'_i . If $u_i \in S^*$ then $S^* \setminus u_i$ is an optimal \mathcal{F} -deletion set of $G'_i \setminus u_i = G_{i+1}$. So if $u_i \in S^*$ then the algorithm succeeds in round i . If S^* is a ρ -cover of G'_i then the probability that $u_i \in S^*$ is at least ρ . Hence the probability that every \mathcal{F} -deletion set of G'_i is a ρ -cover and $u_i \in S^*$ is at least $p = \rho/2$. \square

In each round Algorithm 3 succeeds probability at least p . In a round i where the algorithm succeeds we have that $OPT(G_{i+1}) < OPT(G)$. Since the algorithm terminates when $OPT(G_i) = 0$ we get the following observation.

Observation 4. *There exists a constant $p > 0$ such that the expected number of rounds of a run of Algorithm 3 is at most $\frac{1}{p}OPT(G)$.*

Since the number of rounds where Algorithm 3 fails is at most the total number of rounds it follows from Lemma 15 that the expected size of the output solution $|S|$ is at most $OPT(G) + \frac{1}{p}OPT(G)$. This proves the following lemma.

Lemma 17. *For every connected $\mathcal{F} \in \mathcal{F}$, Algorithm 3 runs in time $O(n + m)$, expected time $O((n + m)OPT(G))$ and outputs an \mathcal{F} solution S with $E[|S|] = c \cdot OPT(G)$ for some constant c .*

While Lemma 10 only gives constant factor approximation algorithms for \mathcal{F} -DELETION for *connected* $\mathcal{F} \in \mathcal{F}$, we can use this approximation algorithm to make an approximation algorithm for all $\mathcal{F} \in \mathcal{F}$.

Theorem 10. *For every $\mathcal{F} \in \mathcal{F}$, \mathcal{F} -DELETION has a constant factor approximation running in time $O(nm)$ and expected time $O((n+m)OPT(G))$. It outputs a feasible solution S with expected size $c \cdot OPT(G)$ for a constant c .*

Proof. By Proposition 1, for every $\mathcal{F} \in \mathcal{F}$ there is a constant η such that for any \mathcal{F} -deletion set S of G we have $\mathbf{tw}(G \setminus S) \leq \eta$. Since TREEWIDTH η -DELETION is a \mathcal{F} -DELETION problem for a connected $\mathcal{F}' \in \mathcal{F}$ it follows from Lemma 10 has a constant factor approximation with the desired running time. We run this algorithm and find a set S' such that $\mathbf{tw}(G \setminus S') \leq \eta$. We have that $E[|S'|] = O(OPT(G))$, where $OPT(G)$ refers to the size of the smallest \mathcal{F} -deletion set in G . Since $\mathbf{tw}(G \setminus S') \leq \eta$ we can solve \mathcal{F} -DELETION on $G \setminus S'$ in linear time and find a set S^* of size $OPT(G \setminus S') \leq OPT(G)$. We return $S = S' \cup S^*$, S is a \mathcal{F} -deletion set of G with expected size $O(OPT(G))$. \square

Interestingly we can also use Algorithm 3 to give a fast randomized FPT algorithm for p - \mathcal{F} -DELETION.

Theorem 11. *For every connected $\mathcal{F} \in \mathcal{F}$, p - \mathcal{F} -DELETION has a randomized $O(c^k n)$ time algorithm. Given a yes instance the algorithm finds a solution and outputs it with probability $1/2$. If the algorithm outputs a solution, it is a feasible solution of size at most k .*

Proof. We modify Algorithm 3 in the following way; if G_{k+1} is not \mathcal{F} -free then output “no” and halt. If the size of the output solution is more than k then output “no” instead. The algorithm runs for at most $k+2$ rounds so the total running time is at most $O(nk)$. If it outputs a solution S then S is an \mathcal{F} deletion of size at most k . We prove that if G has an \mathcal{F} deletion of size at most k then the algorithm will output a solution with probability at least $\frac{1}{c^{k+2}}$ for a constant c . Repeating this algorithm $O((1/c)^k)$ times and outputting a solution if either iteration does then proves the theorem.

In each round, the probability that Algorithm 3 succeeds is at least p for some constant p . Thus the probability that Algorithm 3 succeeds in all its rounds before it terminates (after at most $k+2$ rounds) is at least p^{k+2} . If the algorithm succeeds in all rounds and outputs a solution then this solution is optimal and hence has size at most k if (G, k) is a yes instance. Finally, if the algorithm succeeds for $k+2$ rounds then $OPT(G_1) > OPT(G_2) \dots OPT(G_{k+2})$ and so $OPT(G_1) \geq k+1$. Hence, if (G, k) is a “yes” instance and the Algorithm 3 succeeds in all of its rounds then it will output a solution of size at most k before terminating. This concludes the proof. \square

7 Deterministic Parameterized Algorithms for p - \mathcal{F} -DELETION

We now give a deterministic $O(c^k n \log^2 n)$ time FPT Algorithm for p - \mathcal{F} -DELETION for all connected $\mathcal{F} \in \mathcal{F}$.

Lemma 18. *For every connected $\mathcal{F} \in \mathcal{F}$, there exist constants $\rho, r, s, c < 1$ such that if we run the DFPR with parameters r, s on an instance (G, k) such that G has a \mathcal{F} deletion S which is not a ρ -cover, then the output instance (G', k') satisfies $|V(G)| - |V(G')| \geq \frac{c|V(G)|}{\log |V(G)|}$.*

Proof. If G has a \mathcal{F} -deletion set S' which is not an ρ -cover, it also has a inclusion minimal \mathcal{F} -deletion set S which is not an ρ -cover. Such a minimal S contains no isolated vertices and hence satisfies $N[S] \leq 2 \sum_{v \in S} d(v) \leq 2\rho m$.

By Proposition 1, there exists a constant b such that $\mathbf{tw}(G \setminus S) \leq b$. By Lemma 5, G has a $(4(b+1)|N[S|], 2(b+1))$ -protrusion decomposition. Set $\beta = 2(b+1)$, $r = 3(\beta+1)$ and r' to be the smallest integer such that the protrusion replacer will replace r -protrusions of size at least r' . Set $s = 2^r \cdot r'$. The protrusion decomposition of G is a $(4(b+1)|N[S|], \beta)$ -protrusion decomposition. By Theorem 6 there exist constants $0 < c < 1$ and $0 < \gamma$ such that if we run the DRFPR on G and $4(b+1)|N[S|] \leq \frac{n}{244s}$ then the output graph G' satisfies $|V(G)| - |V(G')| \geq \frac{c|V(G)|}{\log n}$. We show that there is a constant $\rho < \frac{1}{3}$ such that if S is not a ρ -cover, then $|N[S|] \leq \frac{n}{1000(b+1)s}$.

Since $\mathbf{tw}(G \setminus S) \leq b$ we have that $G \setminus S$ is $(b+1)$ -degenerate. If S is not a ρ -cover then $m \leq n(b+1) + \sum_{v \in S} d(v) \leq n(b+1) + 2\rho m$. Rearranging yields that $|N[S|] \leq 2\rho m \leq n \frac{2\rho(b+1)}{1-2\rho} \leq n\rho 6(b+1)$. Choosing $\rho = 6000(b+1)^2 s$ yields that $|N[S|] \leq \frac{n}{1000(b+1)s}$. Hence, if S is not a ρ -cover then the output instance G' of the RFPR satisfies $|V(G)| - |V(G')| \geq \frac{c|V(G)|}{\log n}$. \square

Lemma 19. *For every connected $\mathcal{F} \in \mathcal{F}$ there is an algorithm that given an instance (G, k) , takes $O((n+m)\log^2 n)$ time and outputs an equivalent instance graph (G', k') such that $V(G') \leq V(G)$ and $\text{OPT}(G') \leq \text{OPT}(G)$. Furthermore there exist a constant $0 < \rho < 1$ such that every \mathcal{F} -deletion set S' of G' is a ρ -cover of G .*

Proof. By Lemma 18 there exist constants $\rho, r, s, c < 1$ such that if we run the DFPR with parameters r, s on an instance (G, k) such that G has a \mathcal{F} deletion S which is not a ρ -cover, then the output instance (G', k') satisfies $|V(G)| - |V(G')| \geq \frac{c|V(G)|}{\log |V(G)|}$. We set these constants as guaranteed by Lemma 18.

The algorithm sets $(G_1, k_1) := (G, k)$, $i = 1$ and enters a loop that proceeds as follows. The algorithm runs the DFPR on (G_i, k_i) with parameters r and s , let the output of the DFPR be (G_{i+1}, k_{i+1}) . If $|V(G_i)| - |V(G_{i+1})| < \frac{c|V(G)|}{\log |V(G)|}$ the algorithm halts and outputs G_i . Otherwise, the algorithm increments i and returns to the beginning of the loop.

One iteration of the loop takes time $O((|V(G_i)| + |E(G_i)|) \log |V(G_i)|)$. Furthermore, every $\log n$ consecutive iterations of the loop reduces the number of vertices by a linear fraction. Hence the total running time is bounded by $O(n \log^2 n)$. Let (G', k') be the instance we output. By Lemma 18 we have that every \mathcal{F} -deletion set S' of G' is an ρ -cover of G . \square

We will say that an instance (G, k) is *irreducible* if running the algorithm of Lemma 19 when run on (G, k) just outputs (G, k) unchanged. Observe that if we run the algorithm of Lemma 19 when run on an instance (G, k) , the instance (G', k') output by the algorithm is irreducible. A direct consequence of Lemma 19 is that in an irreducible instance (G, k) every \mathcal{F} -deletion set S in G is a ρ -cover.

We now give a deterministic algorithm for p - \mathcal{F} -DELETION. for connected $\mathcal{F} \in \mathcal{F}$. The intuition behind this algorithm is that vertices of high degree seem more useful for a solution than the vertices of low degree. Towards this we introduce the notion of buckets. We partition the vertex set of G into sets that we refer to as *buckets*, in the following fashion. For every $j \geq 1$ define

$$B_j = \left\{ v \in V(G) \mid \frac{n}{2^j} < d(v) \leq \frac{n}{2^{j-1}} \right\}.$$

We set constants $\eta > 0$ and $d > 0$ such that $\frac{4d+3\eta}{2} < \rho$. For the presentation of the algorithm we fix a \mathcal{F} -deletion set X of size at most k . Next we define a notion of big and good for buckets.

Definition 4. *A bucket B_i is said to be big if $|B_i| > \eta$ and it is said to be good if $|B_i \cap X| \geq d|B_i|$.*

Algorithm-FPT-Det(G, k)

Step 1: Check whether G is \mathcal{F} -free, if yes then **return**(true). Else if $k \leq 0$ and G is not \mathcal{F} -free return that G does not have a k -sized \mathcal{F} -hitting set.

Step 2: Apply Lemma 19 on (G, k) and obtain an equivalent irreducible instance (G^*, k^*) .

Step 3: Let B_j , $j \in \{a, b, \dots, \ell\}$, be the good buckets for G^* . For every good bucket B_j , and for every subset $S \subseteq B_j$ of size at least $d|B_j|$ check whether **Algorithm-FPT-Det**($G^* \setminus \{S\}, k - |S|$) returns true. If any of these calls return true then **return**(true) else **return**(false).

Figure 4: A $2^{O(k)} n \log^2 n$ deterministic FPT algorithm for p - \mathcal{F} -DELETION.

The next lemma says that if (G, k) is a irreducible yes instance to p - \mathcal{F} -DELETION then it has a bucket that is both big and good simulatenously.

Lemma 20. *For any connected $\mathcal{F} \in \mathcal{F}$, let (G, k) be a irreducible yes instance to p - \mathcal{F} -DELETION. Then G has a bucket that is both big and good.*

Proof. Since (G, k) a irreducible yes instance to p - \mathcal{F} -DELETION every optimal \mathcal{F} -hitting set X is a ρ -cover for G , that is, $\sum_{v \in V(G)} d(v) \leq \rho \sum_{v \in X} d(v)$. For a contradiction, assume that G does not have a bucket that is both big and good.

$$\begin{aligned}
 \sum_{v \in X} d(v) &= \sum_{i=1}^{\log n} \sum_{v \in B_i \cap X} d(v) \\
 &= \sum_{\{i | B_i \text{ is not good}\}} \sum_{v \in B_i \cap X} d(v) + \sum_{\{i | B_i \text{ is not big}\}} \sum_{v \in B_i \cap X} d(v) \\
 &\leq d \cdot 4m + \sum_{\{i | B_i \text{ is not big}\}} i\eta \cdot \left(\frac{n}{2^i}\right) \\
 &\leq d \cdot 4m + 3\eta n = 2m \frac{4d + 3\eta}{2} < 2m\rho
 \end{aligned}$$

Which contradicts that X is a ρ -cover. □

Theorem 12. *Let $\mathcal{F} \in \mathcal{F}$ be a connected obstruction set. There exists a deterministic algorithm for p - \mathcal{F} -DELETION running in time $O(c_h^k n \log^2 n)$ on a n vertex graph. The constant c_h only depends on \mathcal{F} .*

Proof. The deterministic algorithm for p - \mathcal{F} -DELETION is described in details in Figure 4. Given a graph G , the algorithm essentially applies Lemma 19 to obtain G^* and then recursively tries to compute the solution to the problem by branching on all large subsets of all the good buckets. The correctness follows directly from Lemma 20. Next we analyze the running time of the algorithm. Suppose for the sake of analysis that all buckets are big, and let a_i be the size of bucket i . Then we have that

$$T(k) \leq \sum_{i=1}^{\log n} \binom{a_i}{k} T(k - da_i)$$

$$T(k) \leq \sum_{i=1}^{\log n} 2^{a_i} T(k - da_i)$$

Assuming $T(k) = x^k$, substitute recursively to get:

$$T(k) \leq \sum_{i=1}^{\log n} 2^{a_i} x^{(k - da_i)}$$

$$T(k) \leq x^k \sum_{i=1}^{\log n} \left(\frac{2}{x^d} \right)^{a_i}$$

If $\frac{2}{x^d} < 1$ then each term of the sum is maximized when the exponent is as small as possible. We will choose x (based on d) such that $\frac{2}{x^d} < 1$ holds. Since $a_i \geq \eta i$ for any big bucket we have that

$$T(k) \leq x^k \sum_{i=1}^{\log n} \left(\frac{2}{x^d} \right)^{\eta i}$$

The sum above is a geometric series and converges to a value that is at most 1 for $x = c$, for a suitably large choice of c depending only on d and η , which depended only on \mathcal{F} . This bounds the running time by c^k . Further, if not all buckets are big the sum above should only be done over the big buckets, yielding the same result. \square

8 Conclusions and open problems

The techniques developed in this paper have several interesting applications. Let us mention a few that we find particularly interesting. Fomin et al. [27] give linear kernels for bidimensional problems on apex-free and H -minor free graphs. The running time of their kernelization algorithms is $O(n^h)$ where h is a constant which depends on the considered graph class. The reason the kernelization algorithms have this running time dependence is that they employ naïve protrusion replacement similarly to the implementation of Lemma 2. If we use the randomized fast protrusion replacer instead we get linear kernels for *all* bidimensional problems from [27] with linear time randomized kernelization algorithms. Using the deterministic fast protrusion replacer yields linear kernels in time $O(n \log^2 n)$.

Most of the problems considered in [27] are CMSO-optimization problems that are strongly monotone. For these problems we can use the *lossless* and fast protrusion replacers to obtain linear kernels that are “lossless”, in the following sense. If G is the original instance and G' is the instance output by the kernelization algorithm, then any feasible solution S' to G' can be lifted (in linear time) to a feasible solution S of G such that $||S| - OPT(G)| \leq ||S'| - OPT(G')|$. This makes the kernelization algorithms combine beautifully with approximation algorithms and heuristics for these problems. For an example combining such lossless kernels with approximation schemes yields $O(n^{O(1)} + f(\epsilon)OPT^{O(1)})$ time approximation schemes for many problems on minor free graphs. When input instances satisfy $OPT \ll n$ but OPT is still too big to run parameterized algorithms, such approximation schemes would be a viable option.

In the framework for obtaining EPTAS on H -minor-free graphs in [25], the running time of the approximation algorithms for many problems is $f(1/\varepsilon) \cdot n^{O(g(H))}$, where g is some function of H only. The only bottleneck for improving the polynomial time dependence in [25] is Lemma 4.1, which gives a constant factor approximation algorithm for TREEWIDTH η -DELETION or η -TRANSVERSAL of running time $n^{O(g(H))}$. Instead of this algorithm we can apply the algorithm from Theorem 1, which runs in time $O(n^2)$. This improves the EPTAS from [25] to run in time $O(f(1/\varepsilon) \cdot n^2)$. For the same reason, the PTAS for many problems on unit disc and map graphs from [26] become EPTAS.

In a companion paper [24] we show that for every \mathcal{F} in \mathcal{F} , p - \mathcal{F} -DELETION admits a polynomial kernel computable in time $O(n^3 \cdot k^c)$ where c is a constant depending only on \mathcal{F} . This yields a deterministic algorithm for p - \mathcal{F} -DELETION with running time $O(2^{O(k \log k)} n^3)$ even for the families $\mathcal{F} \in \mathcal{F}$ that contain disconnected graphs.

An interesting direction for further research is to investigate p - \mathcal{F} -DELETION when none of the graphs in \mathcal{F} are planar. The most interesting case here is when $\mathcal{F} = \{K_5, K_{3,3}\}$ aka the VERTEX PLANARIZATION problem. Surprisingly, we are not aware even of a single case of p - \mathcal{F} -DELETION with \mathcal{F} containing no planar graph admitting either a constant factor approximation, polynomial kernelization, or a parameterized single-exponential time algorithm. It is tempting to conjecture that the line of tractability is determined by whether \mathcal{F} contains a planar graph or not.

Acknowledgements. The authors are grateful to Bart Jansen for insightful discussions, and especially for pointing out that p - \mathcal{F} -DELETION does not have finite integer index when the family \mathcal{F} is not connected.

References

- [1] V. BAFNA, P. BERMAN, AND T. FUJITO, *A 2-approximation algorithm for the undirected feedback vertex set problem*, SIAM Journal on Discrete Mathematics, 12 (1999), pp. 289–297.
- [2] R. BAR-YEHUDA AND S. EVEN, *A linear-time approximation algorithm for the weighted vertex cover problem*, J. Algorithms, 2 (1981), pp. 198–203.
- [3] R. BAR-YEHUDA, D. GEIGER, J. NAOR, AND R. M. ROTH, *Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and Bayesian inference*, SIAM Journal on Computing, 27 (1998), pp. 942–959.
- [4] A. BECKER, R. BAR-YEHUDA, AND D. GEIGER, *Randomized algorithms for the loop cutset problem*, J. Artificial Intelligence Res., 12 (2000), pp. 219–234.
- [5] H. BODLAENDER, *Treewidth: Algorithmic techniques and results*, in MFCS’97: Mathematical Foundations of Computer Science 1997, 22nd International Symposium (MFCS), vol. 1295 of Lecture Notes in Computer Science, Springer, 1997, pp. 19–36.
- [6] H. L. BODLAENDER, *A linear-time algorithm for finding tree-decompositions of small treewidth*, SIAM J. Comput., 25 (1996), pp. 1305–1317.
- [7] H. L. BODLAENDER, F. V. FOMIN, D. LOKSHTANOV, E. PENNINKX, S. SAURABH, AND D. M. THILIKOS, *(meta) Kernelization*, in FOCS 2009, IEEE, 2009, pp. 629–638.

- [8] H. L. BODLAENDER, F. V. FOMIN, D. LOKSHTANOV, E. PENNINKX, S. SAURABH, AND D. M. THILIKOS, *(Meta) kernelization*, in Proc. of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2009, pp. 629–638.
- [9] R. B. BORIE, G. R. PARKER, AND C. A. TOVEY, *Automatic Generation of Linear-Time Algorithms from Predicate Calculus Descriptions of Problems on Recursively Constructed Graph Families*, Algorithmica, 7 (1992), pp. 555–581.
- [10] R. B. BORIE, R. G. PARKER, AND C. A. TOVEY, *Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families*, Algorithmica, 7 (1992), pp. 555–581.
- [11] Y. CAO, J. CHEN, AND Y. LIU, *On feedback vertex set new measure and new structures*, in Proc. of the 12th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2010), vol. 6139 of LNCS, 2010, pp. 93–104.
- [12] J. CHEN, B. CHOR, M. FELLOWS, X. HUANG, D. W. JUEDES, I. A. KANJ, AND G. XIA, *Tight lower bounds for certain parameterized NP-hard problems*, Information and Computation, 201 (2005), pp. 216–231.
- [13] J. CHEN, F. V. FOMIN, Y. LIU, S. LU, AND Y. VILLANGER, *Improved algorithms for feedback vertex set problems*, J. Comput. Syst. Sci., 74 (2008), pp. 1188–1198.
- [14] J. CHEN, I. A. KANJ, AND G. XIA, *Improved upper bounds for vertex cover*, Theor. Comput. Sci., 411 (2010), pp. 3736–3756.
- [15] B. COURCELLE, *The expression of graph properties and graph transformations in monadic second-order logic*, in Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations, G. Rozenberg, ed., World Scientific, 1997, ch. 5.
- [16] B. COURCELLE AND M. MOSBAH, *Monadic second-order evaluations on tree-decomposable graphs*, Theor. Comput. Sci., 109 (1993), pp. 49–82.
- [17] M. CYGAN, J. NEDERLOF, M. PILIPCZUK, M. PILIPCZUK, J. VAN ROOIJ, AND J. O. WOJTASZCZYK, *Solving connectivity problems parameterized by treewidth in single exponential time*, in Proceedings of the 52nd Annual Symposium on Foundations of Computer Science (FOCS 2011), IEEE, 2011, pp. 150–159.
- [18] M. CYGAN, M. PILIPCZUK, M. PILIPCZUK, AND J. O. WOJTASZCZYK, *An improved fpt algorithm and quadratic kernel for pathwidth one vertex deletion*, in IPEC, vol. 6478 of Lecture Notes in Computer Science, 2010, pp. 95–106.
- [19] F. K. H. A. DEHNE, M. R. FELLOWS, M. A. LANGSTON, F. A. ROSAMOND, AND K. STEVENS, *An $O(2^{O(k)}n^3)$ FPT algorithm for the undirected feedback vertex set problem*, Theory Comput. Syst., 41 (2007), pp. 479–492.
- [20] M. R. FELLOWS AND M. A. LANGSTON, *Nonconstructive tools for proving polynomial-time decidability*, J. ACM, 35 (1988), pp. 727–739.
- [21] ———, *An analogue of the myhill-nerode theorem and its use in computing finite-basis characterizations (extended abstract)*, in FOCS, 1989, pp. 520–525.

- [22] S. FIORINI, G. JORET, AND U. PIETROPAOLI, *Hitting diamonds and growing cacti*, in Proceedings of the 14th Conference on Integer Programming and Combinatorial Optimization (IPCO 2010), vol. 6080 of Lecture Notes in Comput. Sci., Springer, 2010, pp. 191–204.
- [23] F. V. FOMIN, D. LOKSHTANOV, N. MISRA, G. PHILIP, AND S. SAURABH, *Hitting forbidden minors: Approximation and kernelization*, in Proceedings of the 8th International Symposium on Theoretical Aspects of Computer Science (STACS 2011), vol. 9 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011, pp. 189–200.
- [24] F. V. FOMIN, D. LOKSHTANOV, N. MISRA, AND S. SAURABH, *PLANAR- \mathcal{F} DELETION: Kernelization and obstruction bounds*. 2012, manuscript.
- [25] F. V. FOMIN, D. LOKSHTANOV, V. RAMAN, AND S. SAURABH, *Bidimensionality and EPTAS*, in Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2011), SIAM, 2011, pp. 748–759.
- [26] F. V. FOMIN, D. LOKSHTANOV, AND S. SAURABH, *Bidimensionality and geometric graphs*, in Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2012), SIAM, 2012, pp. 1563–1575.
- [27] F. V. FOMIN, D. LOKSHTANOV, S. SAURABH, AND D. M. THILIKOS, *Bidimensionality and kernels*, in Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010), 2010, pp. 503–510.
- [28] J. GUO, J. GRAMM, F. HÜFFNER, R. NIEDERMEIER, AND S. WERNICKE, *Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization*, J. Comput. Syst. Sci., 72 (2006), pp. 1386–1396.
- [29] R. IMPAGLIAZZO, R. Paturi, AND F. ZANE, *Which problems have strongly exponential complexity*, Journal of Computer and System Sciences, 63 (2001), pp. 512–530.
- [30] G. JORET, C. PAUL, I. SAU, S. SAURABH, AND S. THOMASSÉ, *Hitting and harvesting pumpkins*, in Proceedings of the 19th Annual European Symposium on Algorithms (ESA2011), vol. 6942 of Lect. Notes Comp. Sc., Springer, 2011, pp. 394–407.
- [31] E. J. KIM, C. PAUL, AND G. PHILIP, *A single-exponential FPT algorithm for K_4 -minor cover problem*. To appear in SWAT 2012.
- [32] A. LANGER, F. REIDL, P. ROSSMANITH, AND S. SIKDAR, *Linear kernels on graphs excluding topological minors*, CoRR, abs/1201.2780 (2012).
- [33] J. M. LEWIS AND M. YANNAKAKIS, *The node-deletion problem for hereditary properties is np-complete*, J. Comput. Syst. Sci., 20 (1980), pp. 219–230.
- [34] C. LUND AND M. YANNAKAKIS, *The approximation of maximum subgraph problems*, in Proceedings of the 20th International Colloquium on Automata, Languages and Programming (ICALP 1993), vol. 700 of Lecture Notes in Comput. Sci., 1993, pp. 40–51.
- [35] M. NAOR, L. J. SCHULMAN, AND A. SRINIVASAN, *Splitters and near-optimal derandomization*, in FOCS, 1995, pp. 182–191.

- [36] G. L. NEMHAUSER AND L. E. TROTTER, JR., *Properties of vertex packing and independence system polyhedra*, Math. Programming, 6 (1974), pp. 48–61.
- [37] G. PHILIP, V. RAMAN, AND Y. VILLANGER, *A quartic kernel for pathwidth-one vertex deletion*, in WG, vol. 6410 of Lecture Notes in Computer Science, 2010, pp. 196–207.
- [38] V. RAMAN, S. SAURABH, AND C. R. SUBRAMANIAN, *Faster fixed parameter tractable algorithms for finding feedback vertex sets*, ACM Transactions on Algorithms, 2 (2006), pp. 403–415.
- [39] B. REED, K. SMITH, AND A. VETTA, *Finding odd cycle transversals*, 32 (2004), pp. 299–301.
- [40] N. ROBERTSON AND P. D. SEYMOUR, *Graph minors. V. Excluding a planar graph*, J. Combin. Theory Ser. B, 41 (1986), pp. 92–114.
- [41] N. ROBERTSON AND P. D. SEYMOUR, *Graph minors. XIII. The disjoint paths problem*, J. Combin. Theory Ser. B, 63 (1995), pp. 65–110.
- [42] M. YANNAKAKIS, *The effect of a connectivity requirement on the complexity of maximum subgraph problems*, J. ACM, 26 (1979), pp. 618–630.
- [43] ———, *Some open problems in approximation*, in Proceedings of the 2nd Italian Conference on Algorithms and Complexity (CIAC 1994), vol. 778 of Lecture Notes in Comput. Sci., 1994, pp. 33–39.